AFRL-RI-RS-TR-2012-162

INTEGRATED ROBUST OPEN-SET SPEAKER IDENTIFICATION SYSTEM (IROSIS)

CARNEGIE MELLON UNIVERSITY

MAY 2012

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2012-162 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/
DARREN M. HADDAD
Work Unit Manager

/s/
WARREN H. DEBANY, JR.
Technical Advisor
Information Grid Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
MAY 2012	Final Technical Report	SEP 2010 – DEC 2011

4. TITLE AND SUBTITLE

Integrated Robust Open-Set Speaker Identification System (IROSIS)

5a. CONTRACT NUMBER
FA8750-10-1-0243

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
35885G

6. AUTHOR(S)

Dr. Qin Jin and Yun Wang

5d. PROJECT NUMBER
ASID

5e. TASK NUMBER
BA

5f. WORK UNIT NUMBER
02

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/Information Directorate
Rome Research Site/RIGC
525 Brooks Road
Rome NY 13441

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2012-162

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report summarizes our effort towards building a robust open-set speaker recognition system. It reviews the various techniques we have used for acoustic feature extraction, speaker modeling, scoring and score normalization, and presents experiment results. We have worked on all the modules of speaker recognition systems. At the front end, we have studied a variety of acoustic features and pre-/post-processing techniques, and have come up with a PPMD feature that combines the benefits of multitaper MFCC, DSCC, pre-emphasis, and short-time feature Gaussianization. At the speaker modeling and scoring stages, we have investigated GMM speaker modeling, SVM speaker modeling, and joint factor analysis (JFA). We have demonstrated that compared to GMM modeling, SVM modeling and scoring are not only better and also faster. We have also shown that T-norm of the scores improves speaker identification performance on the ROSSI database.

15. SUBJECT TERMS
Open-set speaker identification, acoustic feature extraction, speaker modeling, support vector machine (SVM), joint factor analysis (JFA)

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			DARREN M. HADDAD
U	U	U	UU	46	19b. TELEPHONE NUMBER (*Include area code*) N/A

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. SUMMARY

This report summarizes our effort towards building a robust open-set speaker recognition system.

In Section 2, we introduce the structure of speaker recognition systems, the databases we have used in our experiments, and the evaluation criteria we used to evaluate the results.

In Section 3, we review the various techniques we have used for acoustic feature extraction. First we review the acoustic features and pre-/post-processing techniques, and propose a PPMD feature that combines their advantages. Next, we elaborate on a number of speaker modeling and scoring methods, including GMM modeling, SVM modeling, and joint factor analysis (JFA). We also introduce some score normalization techniques.

In Section 4 we present experiment results. The experiments compare the performances of various acoustic features and pre-/post-processing techniques, and show that PPMD achieves a better performance than other features. The experiments also show that SVM speaker modeling is both better and faster than GMM modeling, and T-norm of the scores improves open-set speaker identification performance on the ROSSI database.

The report is concluded in Section 5.

2. INTRODUCTION

2.1 Block Diagram of Speaker Recognition Systems

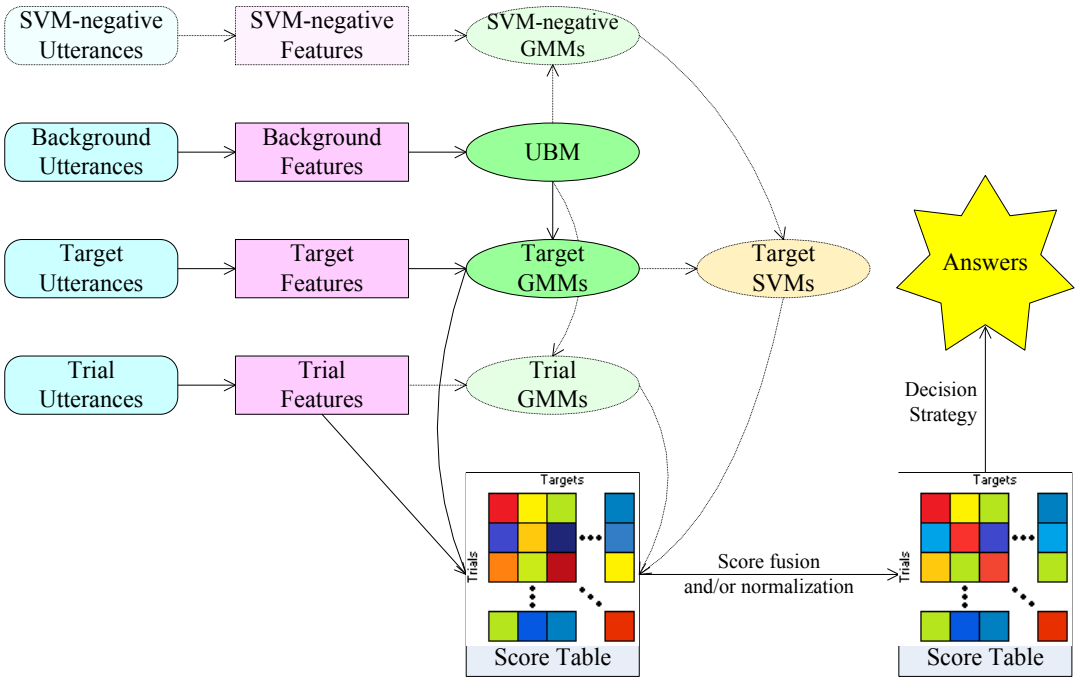

Figure 1. Block Diagram of a Typical Speaker Recognition System (Dotted Blocks and Arrows Denote Modules Special to SVM Speaker Modeling)

Figure 1 is the block diagram of a typical speaker recognition system. A complete run of the system consists of the following steps:

(0) Voice activity detection (VAD). This is a preliminary step in which the speech-containing regions are located in the utterance. Usually people don't spend too much effort here; they usually just apply an energy-based threshold on the frames. We determine the threshold by clustering the log-energy values of the frames in an utterance into 2 clusters using the *k*-means algorithm.

(1) Feature extraction. This turns the speech-containing regions into sequences of feature vectors. We have investigated lots of features (MFCC, MHEC, WMVDR, Multitaper MFCC, SCF & SCM, HSCC, FFV, DSCC) and pre-/post-processing techniques (pre-emphasis, LP, CMN, RASTA, Gaussianization, delta features). Feature fusion also goes here.

(2) UBM training. A universal background model (UBM) is trained from some background utterances. The UBM will be useful in the MAP training during speaker enrollment, and may also act as a reference in scoring.

(3) Speaker enrollment and modeling. The traditional method of modeling target speakers is Gaussian mixture modeling. A Gaussian mixture model (GMM) is either trained independently or adapted from the UBM for each target speaker. Common adaptation techniques include relevance MAP adaptation and joint factor analysis (JFA). It is also possible to model the speakers discriminatively using support vector machines (SVMs), in which case a SVM is trained for each speaker using his/her own GMM as a positive example and the GMMs of many other speakers as negative examples.

(4) Scoring. A score is given for each trial utterance *u* against the model of each candidate speaker *s* for this utterance (or the UBM). The scoring method and its inputs depend on the speaker modeling technique used. For GMM speaker modeling, the score is usually calculated from the target GMM and trial utterance features. This is called "frame-by-frame scoring", and it can be approximated in various ways for faster speed. For SVM speaker modeling, the score is calculated from the target SVM and a GMM trained from the trial utterance.

(5) Score fusion and/or normalization. Normalization techniques include Z-norm, T-norm, ZT-norm and TZ-norm.

(6) Decision. For speaker verification, the score of each trial utterance against its hypothesized speaker is compared against a *global* threshold. For closed-set speaker identification, the target speaker scoring the highest is picked for each trial utterance. For open-set speaker identification, the highest-scoring target speaker is picked, and its score compared against a *global* threshold to decide whether the utterance comes from the target speaker or a non-target speaker.

2.2 Databases Used in the IROSIS Project

The primary database used in this project is the ROSSI database. This is a database for open-set speaker identification, i.e. one has to decide both whether the trial speaker is in the target set and who the speaker is. This database comprises of 8 sets, differing in language, channel, environment, etc. Each set has 100 training utterances (1 for each target speaker) and

200 trial utterances (1 in-set trial utterance for each target speaker and 100 out-of-set trial utterances). In addition to these, Set 1 has 196 development utterances, and Sets 2~8 share 400 development utterances. The development utterances are used for UBM training, SVM training and score normalization.

For some time in the spring of 2011, we also used the MIXER5 database. This database contains speech from 60 male speakers and 81 female speakers. Each speaker speaks in 6 sessions (labeled U ~ Z) of half an hour each, and the speech is recorded in 14 channels (labeled A ~ N). We selected the data to make up four scenarios that are similar to the sets in the ROSSI database. We used Session V for training and Session Y for testing, and mix-matched the channels B (close-talking) and L (far-field). The four scenarios are referred to as VB-YB, VL-YL, VB-YL and VL-YB respectively.

Table 1. Detail of NIST Data Used for Training and Testing

Purpose	Source	No. Speakers	No. Utterances	Duration (hr, voiced part only)
Training UBM	2004, 2005, 2006 single train	692	867	29
Training V	2005, 2006 multi train	435	2908	98
Training D	2004 multi train	125	1382	45
Training U	2005, 2006 TEL test	-	2672	90
Negative examples for SVM	2004 single train	125	246	8
Enrollment	2008 short train	506	648	20
Test	2008 short test	279	620	19

As JFA speaker modeling requires much more data than the ROSSI database could provide, we also used the data from the NIST SRE evaluations in 2004, 2005, 2006, and 2008. The NIST database is for speaker verification, i.e. one only needs to decide whether the trial speaker is the hypothesized target speaker or not. Our experiments have focused on the telephone utterances of male speakers. Table 1 shows how we chose the data for training and testing.

2.3 Evaluation Criteria

In an open-set SID system, the ground truth and the identification result can be classified into five cases:

- Correct accept (CA) – the test speaker is a target speaker, and the system identifies it correctly.
- Speaker confusion error (SE) – the test speaker is a target speaker, but the system misclassifies it as another target speaker.

- False reject (FR, also called "miss") – the test speaker is a target speaker, but the system classifies it as an imposter.
- Correct reject (CR) – the test speaker is an imposter, and the system correctly classifies it as an imposter.
- False accept (FA, also called "false alarm") – the test speaker is an imposter, but the system accepts it as a target speaker.

Evaluation criteria are calculated from the count of the five types of results. We used the following for criteria to evaluate our open-set SID experiments on the ROSSI and MIXER5 databases:

- **Closed-set accuracy**: $\text{AccuClosed} = \dfrac{\text{CA}}{\text{CA}+\text{SE}}$, with the threshold θ set to minus infinity (in which case nothing is rejected, and FR = 0).
- **Open-set accuracy**: The *harmonic mean* of the correct identification rate for I- and O-trials, with the threshold θ set to the value that maximizes the accuracy, which is found by parameter sweeping.

$$\text{AccuOpen} = \max_{\theta} \frac{2}{I^{-1}+O^{-1}}, \qquad I = \frac{\text{CA}}{\text{CA}+\text{SE}+\text{FR}}, \qquad O = \frac{\text{CR}}{\text{CR}+\text{FA}} \tag{1}$$

- **Equal error rate (EER)**: This criterion only evaluates the performance of in/out classification, and considers speaker confusion errors as correct classifications. The EER is the classification error rate when the threshold θ is set so that the false reject rate and false accept rate are identical:

$$\text{EER} = \frac{\text{CA}+\text{SE}}{\text{CA}+\text{SE}+\text{FR}} \overset{\text{Set } \theta \text{ so that}}{=\!=\!=\!=\!=} \frac{\text{CR}}{\text{CR}+\text{FA}} \tag{2}$$

  In order that all the criteria are the bigger the better, we report 1 – EER instead.
- **Precision biased correct decision rate ($F_{0.5}$)**: This criterion also only considers the in/out classification, but is focused on the precision and recall of target trials. It is defined as a *weighted harmonic mean* of the precision and recall. Like the open-set accuracy, the threshold θ is set to maximize the criteria.

$$\text{Precision} = \frac{\text{CA}+\text{SE}}{\text{CA}+\text{SE}+\text{FA}}, \qquad \text{Recall} = \frac{\text{CA}+\text{SE}}{\text{CA}+\text{SE}+\text{FR}} \tag{3}$$

$$F_{0.5} = \max_{\theta} \frac{(1+\beta^2)\cdot\text{Precision}\cdot\text{Recall}}{\beta^2\cdot\text{Precision}+\text{Recall}}, \qquad \beta = 0.5 \tag{4}$$

Closed-set accuracy evaluates only the "identification" performance and doesn't consider the in-set / out-of-set decision; on the other hand, EER and $F_{0.5}$ only evaluate the in/out decision performance. Open-set accuracy is a good overall measure of the performance.

In a speaker verification task, there are no speaker confusion errors (SE). We use the equal error rate (EER) as the criterion to evaluate our speaker verification experiments on the NIST database.

3. METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 Acoustic Feature Extraction

3.1.1 Various Acoustic Features

(0) Baseline MFCC feature. The procedure of extracting the baseline MFCC feature is shown in Figure 2. Based on some pilot experiments on the MIXER5 experiment, we chose to use CMN (cepstral mean normalization) as the normalization technique (in which we normalize both the mean and the variance of the features). We didn't use RASTA filtering [1] or spectral subtraction for noise reduction as they didn't contribute to the performance.

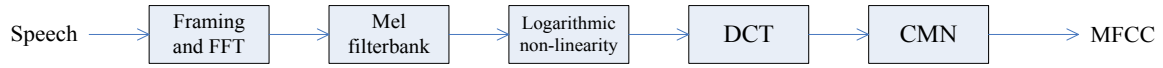

Figure 2. Procedure of Extracting the Baseline MFCC Feature

(1) MHEC (mean Hilbert envelop coefficients) [2]. The procedure of extracting the MHEC feature is shown in Figure 3. It is claimed to be magically robust to car noise. However, the extraction procedure is not very different from MFCC: the Gammatone filterbank is similar to the Mel filterbank although the former is in the time domain and the latter is in the frequency domain; the calculation of the Hilbert envelope is similar to that of the energy in each time-frequency unit; and the long-term average is similar to CMS (cepstral mean subtraction). There seems to be no theoretical foundation why MHEC should be robust to car noise, and our experiments didn't support this claim, either.

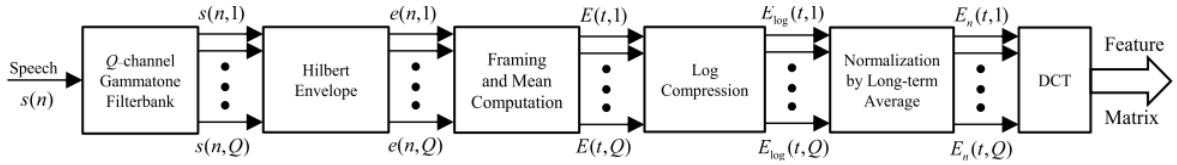

Figure 3. Procedure of Extracting the MHEC Feature

(2) WMVDR (warped minimum variance distortionless response) [3]. MVDR is a method for estimating a smoothed version of a signal's power spectrum. Its warped version, warped MVDR (WMVDR), is used to replace the FFT and filterbank steps in the extraction of MFCC, producing a smoother power spectrum with different resolution at different frequencies just like the output of the Mel filterbank. There had been experiments [3] showing that WMVDR features perform better than MFCC features. But our experiments have shown that the performance of WMVDR features wasn't very different from MFCC.

This means that a smoother power spectrum doesn't have much effect on the final identification performance.

(3) Multitaper MFCC [4]. The motivation of multitaper MFCC is to reduce the variance in the power spectrum estimation. It estimates the power spectrum of a frame using a series of different windows (also called "tapers"), and then average the different estimations of the power spectrum. The operation of averaging reduces the variance in the power spectrum at the cost of reduced resolution, but resolution is not important because the power spectrum will be fed into the Mel filterbank anyway. The three types of multitapers introduced in [4] all resulted in significant improvement in closed-set accuracy on the ROSSI database, especially the multipeak series with 8 windows.

(4) SCF & SCM (spectrum centroid frequency & magnitude) [5]. In the computation of MFCC, the energy distribution within each band is ignored. SCF aims to incorporate this information in the feature as well. Defined as the average of the frequency points on the spectrum within a band weighted by the magnitude at these frequency points, SCF describes how much the energy distribution deviates from the center frequency in each band. SCM is defined as the average of the magnitudes within a band weighted by frequency, and can be computed efficiently together with SCF. Although there's no reason to weight the magnitude by frequency, because the bands are narrow, SCM is actually almost equal to the MFCC.

Since SCF contains information that is absent in MFCC, a fusion of the two features is expected to improve the performance. However, the performance of SCF alone turned out so poor that neither feature fusion nor score fusion performed as well as MFCC.

(5) HSCC & FFV (harmonic structure cepstral coefficients & fundamental frequency variation) [6][7]. These are two prosodic features that provide complementary information to MFCC. HSCC models the probabilistic distribution of the fundamental frequency by harmonic summation, and FFV models the rate of variation of the fundamental frequency by stretching and comparing the short-time spectrum of adjacent frames. We carried out score fusion of MFCC, HSCC and FFV on the MIXER5 database. The fusion resulted in marginal improvement in closed-set accuracy, but the fusion weights were quite counter-intuitive (FFV often carries a larger weight than MFCC).

(6) DSCC (delta-spectral cepstral coefficients) [8]. This feature aims at enhancing the noise robustness of delta MFCC features. Its extraction procedure is compared with that of MFCC in Figure 4. In extracting delta MFCC, the differential is performed last, long after the logarithm. Additive noise can have a destructive effect on the log spectrum filling in all the deep valleys, therefore the differential of the clean log spectrum and noisy log spectrum will be very different. But on the spectrum before logarithm, the effect of additive noise is relatively small. DSCC calculates the differential before applying the logarithmic non-linearity, and since the differential can have negative values, the logarithm is replaced by Gaussianization. Our experiments have shown that DSCC is more robust to noise than delta MFCC.

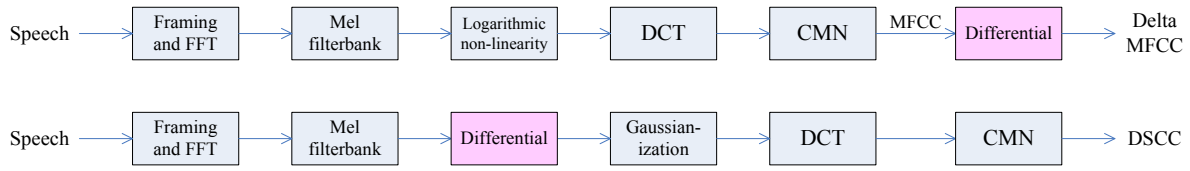

Figure 4. Comparison of the Extraction of the Delta MFCC and DSCC Features

3.1.2 Pre- and Post-Processing Techniques

(1) Pre-emphasis and delta features. These are traditional pre- and post-processing techniques. We have verified that they do improve the performance on the ROSSI database, both closed-set and open-set.

(2) Linear prediction (LP) pre-processing for noise reduction. Linear prediction can be used for speech coding, because speech signals have patterns and are largely predictable. On the other hand, noise is random and unpredictable. This inspired us to think that pre-processing a noisy signal with linear prediction can increase the signal-to-noise ratio (SNR) of the signal. We applied LP with block filtering up to 20^{th} order; later we discovered some related work [9] that used least-mean-squares (LMS) filtering up to several hundredth order. Most combinations of filtering method and filter order were confirmed to increase the SNR, and the typical SNR gain is between 5 ~ 10 dB. However, the increase in SNR didn't translate directly into improved performance of the back-end system. We experimented with three back-end systems (speaker recognition on the MIXER5 database, continuous spoken digits recognition, and HMM-based pitch tracking), and the performance of all the three systems stayed the same no matter we applied LP pre-processing or not.

(3) Short-time feature Gaussianization [10]. This is an advanced version of CMN. Gaussianization not only normalizes the mean and variance of the features, but warps them non-linearly to conform to the Gaussian density function. In short-time feature Gaussianization, a feature value that is the *k*-th smallest in an *N*-point window around it will be warped to $\Phi^{-1}[(k-1)/N]$, where Φ is the Gaussian cumulative density function (CDF). Short-time feature Gaussianization has shown consistent performance improvement across most conditions, although there hasn't been any determinative explanation as to why. A plausible explanation is that a Gaussian distribution of the features agrees best with the GMM speaker modeling.

3.1.3 Best Feature Selected

Our experiments with the various types of acoustic features and pre-/post-processing techniques (see Section 4.1) showed the following to be beneficial: Multitaper MFCC (with 8 multipeak tapers), DSCC, pre-emphasis, short-time feature Gaussianization. We combined these all and arrive at a 40-dimensional acoustic feature whose extraction procedure is shown in Figure 5. According to our old naming conventions, this feature should be called

PRE_**P**EAK8_**M**FCC20_GAUSS300 + PRE_PEAK8_**D**SCC20(GAUSS300)_CMN
And we call it PPMD for short.

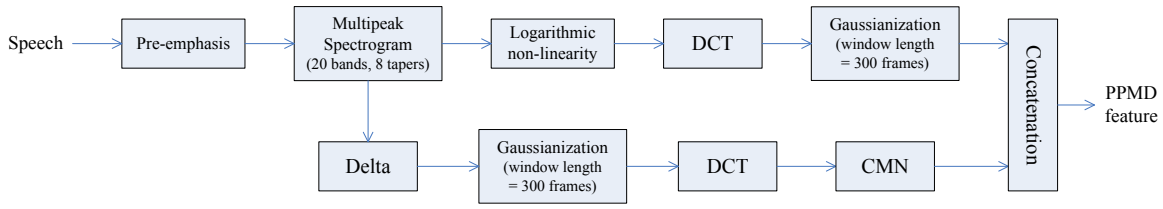

Figure 5. Procedure of Extracting PPMD Features

In addition to this, we also used a version of MFCC features that was used previously in other projects. This feature also has 40 dimensions, and we call it RM40. RM40 shows slightly inferior performance to PPMD on the ROSSI database, but is consistently better than PPMD on the NIST database. This verifies the fact that there is no universal best feature.

3.2 Speaker Modeling and Scoring

3.2.1 Gaussian Mixture Modeling (GMM)

The basic method of modeling the speech of a target speaker is Gaussian mixture models (GMM). A GMM is made up of C components (in our experiments, C = 256 for ROSSI and MIXER5, and C = 1024 for NIST), each component i having a mean vector μ_i, a (typically diagonal) covariance matrix Σ_i, and a weight w_i.

The GMM for a target speaker can be trained entirely from the data of this speaker, but it is usually preferred to adapt it from the UBM. There are two reasons to favor adaptation: there may be not enough data to train a speaker GMM robustly, while UBM can provide a reasonable prior; the components of an adapted GMM are aligned with those of the UBM, which is crucial for the SVM modeling to be introduced. In adaptation, usually the covariance matrices and weights are kept fixed, and only the component means are adapted.

3.2.1.1 GMM Adaptation Techniques

Common GMM adaptation techniques include relevance MAP adaptation [11] and joint factor analysis (JFA).

In relevance MAP adaptation, first an EM iteration is run with the UBM as the initial value to obtain a set of μ_i's. Then these μ_i's are interpolated with those of the UBM via a relevance factor r:

$$\mu_i \leftarrow \frac{n_i \mu_i + r \mu_i^{UBM}}{n_i + r} \tag{5}$$

where n_i is the soft count of feature vectors aligned with component *i*. This interpolation has the effect that the less data a component has, the more its mean is pulled back to the UBM mean. The appropriate value of *r* depends on the amount of data available, but the performance is found to be not very sensitive to *r*. A typical value of *r* is 16.

Relevance MAP adaptation is called "MAP" because it is equivalent to imposing a prior on the component means:

$$\mu_i \sim \mathsf{N}\,(\mu_i^{UBM}, \frac{1}{r}\Sigma_i) \tag{6}$$

Joint factor analysis is a more sophisticated version of MAP adaptation, where the parameters of the prior also need to be trained from data. Due to its complexity, it deserves separate explanation in Section 3.3.

3.2.1.2 Scoring Methods with GMM Modeling

The basic method to score an utterance against a GMM is full frame-by-frame scoring. Denote the feature vectors of the utterance by $X = \{X_1, \ldots, X_n\}$, and the GMM by $G = \{\mu_i, \Sigma_i, w_i\}$, then the score is defined as the frame-average log-likelihood:

$$\text{score}(X, G) = \frac{1}{n}\sum_{t=1}^{n} \log \sum_{i=1}^{C} w_i\, p(X_t; \mu_i, \Sigma_i) \tag{7}$$

where $p(X; \mu, \Sigma)$ is the multivariate Gaussian density function. This score is usually adjusted by subtracting the score of the trial utterance against the UBM.

Full frame-by-frame scoring can be slow due to the summation across the GMM components. When an utterance needs to be scored against multiple GMMs (e.g. in speaker identification, or in speaker verification where an utterance needs to be verified against multiple speakers), it can be beneficial to use fast frame-by-frame scoring ([11],). In fast frame-by-frame scoring, the second summation doesn't go across all components, but only a few (say, 5) top components:

$$\text{score}(X, G) = \frac{1}{n}\sum_{t=1}^{n} \log \sum_{i \in TC_t} w_i\, p(X_t; \mu_i, \Sigma_i) \tag{8}$$

The "top components" set TC_t is chosen for each feature vector based on the UBM: it contains the components that yield the several highest values of $w_i\, p(X_t; \mu_i^{UBM}, \Sigma_i)$.

3.2.2 Support Vector Machine (SVM) Modeling

GMM is a *generative* modeling technique of speakers, but speakers can also be modeled *discriminatively* with support vector machines (SVM). For each target speaker, a GMM is adapted from the UBM for each training utterance, and these are used as the positive training examples for the SVM. In addition, a GMM is adapted from the UBM for each utterance in a pool of "SVM-negative" utterances, and these are used as negative training examples for the SVMs of all target speakers. It is possible to re-use the utterances used for UBM training as SVM-negative utterances.

To train an SVM with GMMs as training examples, it is necessary to come up with a kernel function for GMMs. Because the component covariances and weights are not adapted, a GMM can also be represented by the collection of its component means. The *CF*-dimensional vector obtained by stacking the *F*-dimensional mean vectors of the *C* components is called the supervector. A kernel function for GMMs is an inner product of the supervectors, optionally with some normalization and weighting with the covariance matrices and component weights.

We used the following six types of kernel functions:

$$K_{\mathrm{PLAIN}}(M_a, M_b) = \sum_i (\mu_i^a)^T (\mu_i^b) \tag{9}$$

$$K_{\mathrm{GUMI}}(M_a, M_b) = \sum_i (\mu_i^a)^T \Sigma_i^{-1} (\mu_i^b) \tag{10}$$

$$K_{\mathrm{KL}}(M_a, M_b) = \sum_i w_i (\mu_i^a)^T \Sigma_i^{-1} (\mu_i^b) \tag{11}$$

$$K_{\mathrm{L2}}(M_a, M_b) = \sum_i \frac{w_i^2}{|\Sigma_i|^{1/2}} \exp\left[-\frac{(\mu_i^a - \mu_i^b)^T \Sigma_i^{-1} (\mu_i^a - \mu_i^b)}{4} \right] \tag{12}$$

$$K_{\mathrm{BHATT}}(M_a, M_b) = \sum_i \exp\left[-\frac{(\mu_i^a - \mu_i^b)^T \Sigma_i^{-1} (\mu_i^a - \mu_i^b)}{8} \right] \tag{13}$$

$$K_{\mathrm{WBHATT}}(M_a, M_b) = \sum_i w_i \exp\left[-\frac{(\mu_i^a - \mu_i^b)^T \Sigma_i^{-1} (\mu_i^a - \mu_i^b)}{8} \right] \tag{14}$$

These functions are derived from different motivations; see A for details. Despite their different motivations, the forms of the kernels look similar, and it is possible to tweak the powers and constants in the formulas to make new kernels. Our experience (see Section 4.2 for example), however, consistently shows that there's not much difference between the performances of five of these kernels, except for L2 which performs significantly worse.

To score a trial utterance against the SVM of target speaker, first adapt a GMM for this utterance, then substitute this GMM into the discriminant function of the SVM. For a negative trial utterance, the score is usually around -1.0. However, because the SVMs are usually trained with very few positive examples (often only one), the score can be negative even for positive trial utterances. Our experience is that the scores for positive trial utterances are usually around -0.7. As a result, we cannot directly perform speaker verification based on the sign of this score, but need to tune the threshold on development data.

Our experiments on both the ROSSI database (see Section 4.2) and the NIST database (see Section 4.3) have shown that SVM modeling is significantly better than GMM modeling.

3.3 Joint Factor Analysis (JFA) Modeling

JFA is a generalization of relevance MAP adaptation in GMM modeling. It assumes that the supervector *m* of a speaker GMM can be decomposed as

$$m = M + Vy + Ux + Dz \tag{15}$$

In this formula, *M* is the UBM supervector; *V* and *U* are tall and thin matrices with *CF* rows and only a few hundred columns whose columns are **eigenvoices** and **eigenchannels**; *D* is a *CF*-by-*CF* diagonal matrix; *y*, *x*, *z* are random vectors that have sizes compatible with the matrices, and obey the standard normal distribution. In other words, it imposes on the speaker GMM supervector *m* a prior distribution of the following form:

$$m \sim \mathsf{N}\,(M, VV^T + UU^T + D^2) \tag{16}$$

It can be shown that relevance MAP adaptation is a special case of JFA with $V = U = 0$ and $D^2 = \Sigma / r$ (Σ is a diagonal block matrix whose blocks are the component covariance matrices).

In Eq. (15), the terms *Vy* and *Dz* are considered to come from the speaker characteristics, while the term *Ux* is considered to arise due to environment and channel variations. Therefore the latter term is dropped, and the supervector is take to be the sum $M + Vy + Dz$.

To obtain GMM models (or supervectors) for a set of speakers via JFA, there are two major steps. First, the matrices *V*, *U*, *D* specifying the prior distribution must be trained from data. Second, the vectors *y*, *x*, *z* need to be estimated under the MAP criterion. Both these steps involve complicated mathematics; we give a tutorial in Section 3.3.1.

As an extension of relevance MAP adaptation, JFA also falls in the category of GMM speaker modeling, and therefore can be used in combination with SVM speaker modeling. It was our goal to show that JFA + SVM modeling would be better than relevance MAP + SVM, but unfortunately we weren't able to achieve this goal.

JFA was first proposed in a fully convolved form [12], and was later simplified by multiple researchers, resulting in many variants. These variants and our choice will be introduced in Section 3.3.2. In order to deal with the channel mismatch between the target GMM (where the channel effect *Ux* has been dropped) and the trial utterance, many scoring methods have been proposed for JFA [13], with different levels of approximation. These scoring methods will be introduced in Section 3.3.3. Our experiment setup and results of JFA will be given in Section 4.3.

3.3.1 A Tutorial of JFA

3.3.1.1 ML training of GMMs, and Baum-Welch statistics

To fully understand the mathematics involved in JFA, it is helpful to review the maximum likelihood (ML) training of GMMs.

Suppose we want to train a GMM with C components from a set of F-dimensional - feature vectors $X = \{X_1, \ldots, X_T\}$ (all column vectors). For the c-th component ($c = 1, \ldots, C$), we denote the mean vector by m_c (column vector), the covariance matrix by Σ_c, and the component weight by w_c. We assume the covariance matrices to be diagonal. For conciseness, we define the CF-dimensional "**supervector**" m as the vertical concatenation of the mean vectors, and the $CF \times CF$ "supercovariance" matrix Σ as the diagonal concatenation of the covariance matrices.

In our problem, we assume the covariance matrices and the components to be fixed and known, and we need to find the supervector m that best fits the data X. Under the maximum likelihood (ML) criterion, we seek to maximize the log likelihood of the data X given the supervector m:

$$\begin{aligned} L(m) &= \log P(X \mid m) = \sum_t \log P(X_t \mid m) = \sum_t \log \sum_{Z_t} P(Z_t) P(X_t \mid Z_t, m) \\ &= \sum_t \log \sum_c \frac{w_c}{(2\pi)^{F/2} \mid \Sigma_c \mid^{1/2}} \exp\left[-\frac{1}{2} (X_t - m_c)^T \Sigma_c^{-1} (X_t - m_c) \right] \end{aligned} \tag{17}$$

where $Z_t = 1, \ldots, C$ refers to the index of the component that generated the data sample X_t. This objective function is hard to optimize directly because of the sum nested in the log, and the usual solution is the EM algorithm. The EM algorithm requires an auxiliary function $Q(m, m')$ that satisfies

$$L(m) = Q(m, m), \qquad Q(m, m') \le L(m') \tag{18}$$

In each EM iteration, the auxiliary function $Q(m, m')$ is maximized w.r.t. m', and the old supervector m is replaced by the new supervector m'. Eq. (18) guarantees that the objective function $L(m)$ is non-decreasing.

The expression of the auxiliary function is given by:

$$
\begin{aligned}
Q(m,m') &= L(m') - \sum_t \mathrm{KL}(Z_t \mid X_t, m \,\|\, Z_t \mid X_t, m') \\
&= \sum_t \log P(X_t \mid m') - \sum_t \sum_{Z_t} P(Z_t \mid X_t, m) \log \frac{P(Z_t \mid X_t, m)}{P(Z_t \mid X_t, m')} \\
&= \sum_t \sum_{Z_t} P(Z_t \mid X_t, m) \left[\log P(X_t \mid m') - \log \frac{P(Z_t \mid X_t, m)}{P(Z_t \mid X_t, m')} \right] \qquad (19) \\
&= \sum_t \sum_{Z_t} P(Z_t \mid X_t, m) \log \frac{P(X_t, Z_t \mid m')}{P(Z_t \mid X_t, m)} \\
&= \sum_t \sum_{Z_t} P(Z_t \mid X_t, m) \log P(X_t, Z_t \mid m') + \sum_t H(Z_t \mid X_t, m)
\end{aligned}
$$

The second term (entropy) is a "constant" that does not depend on *m*'.

In the E-step of an EM iteration, we calculate the posterior probability of the component index given the data $P(Z_t \mid X_t, m)$, according to the old supervector *m*. This step is also called aligning the data with the components. We denote $P(Z_t = c \mid X_t, m)$ by N_{tc} for short, and define the following statistics (called Baum-Welch statistics [14]) of the data *X* for each component, which will be useful in the M-step:

- Zeroth order statistics $N_c = \sum_t N_{tc}$: the "number" of data samples aligned with the *c*-th component.
- First order statistics $F_c = \sum_t N_{tc} X_t$: the sum of the data samples aligned with the *c*-th component.
- Second order statistics $S_c = \sum_t N_{tc} X_t X_t^T$: the sum of the "squares" of the data samples aligned with the *c*-th component. This will not be useful since we're only training the mean vectors of the GMM.

We also define a *CF* × *CF* diagonal matrix *N* whose diagonal blocks are $N_c I_F$, where $c = 1, \ldots, C$ and I_F is a unit matrix of order *F*, and a *CF*-dimensional column vector *F* as a vertical concatenation of the F_c for all the components. It's worth pointing out that, although not explicitly noted, the Baum-Welch statistics (including *N* and *F*) are functions of the data *X* and the supervector *m*.

Now we look at the M-step, in which we maximize $Q(m,m')$ w.r.t. *m*'. We simplify $Q(m,m')$ as follows:

$$
\begin{aligned}
Q(m,m') &= \text{constant} + \sum_t \sum_{Z_t} P(Z_t \mid X_t, m) \log P(X_t, Z_t \mid m') \\
&= \text{constant} + \sum_t \sum_c N_{tc} \left[\log \frac{w_c}{(2\pi)^{F/2} \mid \Sigma_c \mid^{1/2}} - \frac{1}{2} (X_t - m'_c)^T \Sigma_c^{-1} (X_t - m'_c) \right] \\
&= \text{constant} + \sum_t \sum_c N_{tc} (m'^T_c \Sigma_c^{-1} X_t - \frac{1}{2} m'^T_c \Sigma_c^{-1} m'_c) \\
&= \text{constant} + m'^T \Sigma^{-1} F - \frac{1}{2} m'^T \Sigma^{-1} N m'
\end{aligned}
\tag{20}
$$

Setting $\nabla_{m'} Q(m,m') = \Sigma^{-1} F - \Sigma^{-1} N m' = 0$, we obtain the optimal new supervector:

$$\hat{m} = N^{-1} F \tag{21}$$

which is the familiar conclusion of setting the new mean vector of each component as the average of the data aligned with that component.

The entire EM algorithm starts by initializing the supervector *m* randomly or using *k*-means clustering, and then repeats the EM iteration for a given number of times or until convergence. The Baum-Welch statistics are re-calculated in every iteration.

3.3.1.2 MAP adaptation of GMMs in general

Sometimes we don't have enough data to train a GMM with ML robustly. In such cases we can train the GMM with MAP, i.e. imposing a prior distribution on the supervector *m*. The most common prior is the Gaussian $\mathsf{N}(M, Z)$, where *M* is typically the supervector of the UBM. The MAP objective function is:

$$L'(m) = L(m) - \frac{1}{2} (m - M)^T Z^{-1} (m - M) \tag{22}$$

Similarly, the auxiliary function will also include the prior term:

$$Q'(m,m') = \text{constant} + m'^T \Sigma^{-1} F - \frac{1}{2} m'^T \Sigma^{-1} N m' - \frac{1}{2} (m' - M)^T Z^{-1} (m' - M) \tag{23}$$

Setting $\nabla_{m'} Q'(m,m') = \Sigma^{-1} F - \Sigma^{-1} N m' - Z^{-1} (m' - M) = 0$, we obtain the optimal new supervector:

$$\hat{m} = (I + Z\Sigma^{-1} N)^{-1} (M + Z\Sigma^{-1} F) \tag{24}$$

Because calculating the Baum-Welch statistics can be time-consuming, in MAP adaptation we usually perform only one iteration, initializing the supervector to be that of the UBM. Therefore the Baum-Welch statistics are also calculated against the UBM.

Now is a good chance to understand why the relevance MAP adaptation is called MAP. Let the covariance matrix of the prior distribution be

$$Z = \frac{1}{r} \Sigma \tag{25}$$

where *r* is the relevance factor. It is easy to see that the adapted supervector will be

$$\hat{m} = (rI + N)^{-1}(rM + F) \tag{26}$$

which is exactly the rule of relevance adaptation.

3.3.1.3 GMM adaptation using JFA

JFA is just a way of specifying the prior distribution of the supervector *m*. It formulates *m* as

$$m = M + Ux + Vy + Dz \tag{27}$$

In this formula, *M* is the UBM supervector. *U* and *V* are tall-and-thin matrices with *CF* rows (typically between 10^4 and 10^5) but only several hundred columns, and *D* is a *CF* × *CF* diagonal matrix. *x*, *y*, *z* are mutually independent random column vectors compatible with the size of *U*, *V*, *D*, and obeying the standard Gaussian distribution $\mathsf{N}(0,1)$. The term *Ux* models the variability in the supervector due to the channel, and *Vy* + *D*z models the variability due to the speaker. The columns of *U* and *V* act as eigenvectors, and are called eigenchannels and eigenvoices respectively. The elements of *x* and *y* are called channel factors and speaker factor[1].

Eq. (27) effectively specifies the following prior distribution:

$$m \sim \mathsf{N}(M, UU^T + VV^T + D^2) \tag{28}$$

It seems we can substitute $Z = UU^T + VV^T + D^2$ into Eq. (24), and the problem is solved.

But there are two issues:

(1) Calculating Eq. (24) involves inverting a *CF* × *CF* non-diagonal matrix, which is computationally intractable;

(2) The channel variability modeled by the *Ux* term is actually a nuisance that we wouldn't like to include in the updated supervector, i.e. we want $\hat{m} = M + V\hat{y} + D\hat{z}$. This requires us to explicitly solve for the optimal adapted values of *x*, *y*, and *z*.

In order to do this, we can write the MAP auxiliary function in terms of *x*, *y*, *z* and *x*', *y*', *z*':

$$\begin{aligned} Q'(x, y, z, x', y', z') = \text{constant} &+ (M + Ux' + Vy' + Dz')^T \Sigma^{-1} F \\ &- \frac{1}{2}(M + Ux' + Vy' + Dz')^T \Sigma^{-1} N (M + Ux' + Vy' + Dz') \\ &- \frac{1}{2} x'^T x' - \frac{1}{2} y'^T y' - \frac{1}{2} z'^T z' \end{aligned} \tag{29}$$

In a more general version of the EM algorithm, we don't need to *maximize* the auxiliary function in the M-step; we only need to *increase* it. Therefore we take a two-step approach following the code in [15]: in the first step we find *x*' and *y*' to maximize *Q*' assuming *z*' = 0,

[1] The terminology is not unified. In some literature, the columns of *U* and *V* are called **channel factors** and **speaker factors**, while the elements of *x* and *y* are called **factor loadings**.

and in the second step we find z' to maximize Q' assuming the values x' and y' found in the last step. Because $Ux+Vy=[U\ V][x;y]$ (where the semicolon stands for vertical concatenation), the method for solving for x' and y' jointly is identical to that for solving y' alone assuming x' = 0. We absorb everything in Eq. (29) that doesn't depend on y' into the constant:

$$Q'(y,y') = \text{constant} + y'^T V^T \Sigma^{-1}(F - NM) - \frac{1}{2} y'^T (I + V^T \Sigma^{-1} NV) y' \tag{30}$$

Setting $\nabla_{y'} Q'(y,y') = V^T \Sigma^{-1}(F - NM) - (I + V^T \Sigma^{-1} NV) y' = 0$, we obtain the optimal new speaker factors:

$$\hat{y} = (I + V^T \Sigma^{-1} NV)^{-1} V^T \Sigma^{-1}(F - NM) \tag{31}$$

If we solved for x and y jointly, then the optimal new factors would be:

$$[\hat{x};\hat{y}] = (I + [U\ V]^T \Sigma^{-1} N [U\ V])^{-1} [U\ V]^T \Sigma^{-1}(F - NM) \tag{32}$$

The matrix inversion here is no challenge because the order of the matrix is only several hundred.

By letting $x' = \hat{x},\ y' = \hat{y}$ and setting the gradient of Eq. (29) w.r.t. z' to 0, we can obtain the optimal z':

$$\hat{z} = (I + D^2 \Sigma^{-1} N)^{-1} D \Sigma^{-1} [F - N(M + U\hat{x} + V\hat{y})] \tag{33}$$

The matrix inversion here is also easy because the matrix to invert is diagonal.

As said in the previous subsection, in MAP adaptation we only do one EM iteration, initializing the supervector to that of the UBM. Therefore the Baum-Welch statistics N and F in Eqs. (32) and (33) are calculated against the UBM. Since we do not want to include the channel variability in the adapted GMM, the adapted supervector will be

$$\hat{m} = M + V\hat{y} + D\hat{z} \tag{34}$$

3.3.1.4 Training the matrices U, V, D

The procedure described in the last subsection happens in the "speaker enrollment" step, where the matrices U, V, D are known, and the training data X are the feature vectors extracted from the utterance(s) of each speaker. However, the matrices U, V, D need to be trained before enrolling the speakers, and this problem is dealt with in this subsection.

We train these matrices one by one in the order of V, U, D, following the code in [15]. When training one matrix, we assume that the matrices not yet trained are equal to zero. We don't train the matrices jointly because it would be mathematically much more difficult, and also because it is pointed out in [16] that when they're trained jointly, D often gets undertrained – most of the speaker variability is accounted for by Vy, and Dz has little effect. This is probably because V has many more degrees of freedom than D.

Before describing the training algorithm, which is explained in detail in [17], we shall

point out an implicit approximation made by that paper on the data likelihood $P(X \mid m)$. From Eq. (18), we know that

$$\log P(X \mid m) = L(m) \geq Q(M, m) \tag{35}$$

where M is the UBM supervector, and that the difference between $L(m)$ and $Q(M, m)$ is the Kullback-Leibler divergence between the "alignment" of the data against the UBM and the GMM with supervector m. We assume the divergence is small enough to neglect, thereby approximating the data likelihood by

$$\log P(X \mid m) \approx Q(M, m) \tag{36}$$

Then, from Eq. (20), we have

$$\log P(X \mid m) \approx \text{constant} + m^T \Sigma^{-1} F - \frac{1}{2} m^T \Sigma^{-1} N m \tag{37}$$

where the Baum-Welch statistics N and F are calculated against the UBM. Applying this approximation has the advantage that the Baum-Welch statistics can be pre-calculated against the UBM and stored, and needn't be re-calculated the in the iterative procedure of training U, V, D.

(1) Training the *V* matrix

Training the V matrix requires utterances from multiple speakers, and ideally multiple utterances from each speaker to average out the channel variability. Denote by $X(s)$ the feature vectors of the speaker s. The training objective of the V matrix is to maximize the total data likelihood:

$$L(V) = \sum_s \log P(X(s) \mid V) \tag{38}$$

The conditioned probability may look weird, but we can plug in the hidden variable $y(s)$, just like Z_t in Eq. (17):

$$L(V) = \sum_s \log \int_{y(s)} P(y(s)) \cdot P(X(s) \mid y(s), V) \, \mathrm{d}y(s) \tag{39}$$

Again, we're having an integral nested within a log, which we tackle with the EM algorithm. The auxiliary function is:

$$\begin{aligned} Q(V, V') &= \text{constant} + \sum_s \int_{y(s)} P(y(s) \mid X(s), V) \log P(X(s), y(s) \mid V') \, \mathrm{d}y(s) \\ &= \text{constant} + \sum_s \int_{y(s)} P(y(s) \mid X(s), V) \log P(X(s) \mid y(s), V') \, \mathrm{d}y(s) \end{aligned} \tag{40}$$

For clarity, we will drop the argument (s) hereafter. The $\int_y P(y \mid X, V)[\cdot] \, \mathrm{d}y$ part is actually an expectation operator over the posterior distribution of y given the data X, which we will denote as E for short. With the approximation of Eq. (36), this posterior distribution is given

by Eq. (30), i.e.

$$y \sim \mathsf{N}\left((I + V^T\Sigma^{-1}NV)^{-1}V^T\Sigma^{-1}(F - NM),\ (I + V^T\Sigma^{-1}NV)^{-1}\right) \tag{41}$$

where the Baum-Welch statistics are calculated against the UBM. From this we can calculate some statistics about *y*:

$$Ey = (I + V^T\Sigma^{-1}NV)^{-1}V^T\Sigma^{-1}(F - NM) \tag{42}$$

$$E(yy^T) = (Ey)(Ey)^T + (I + V^T\Sigma^{-1}NV)^{-1} \tag{43}$$

which will be useful in simplifying $Q(V, V')$.

Now let's go on simplifying Eq. (40). The part $\log P(X \mid y, V')$ is approximated by Eq. (37):

$$\begin{aligned} Q(V,V') &= \text{constant} + \sum_s \left\{ E((M + V'y)^T\Sigma^{-1}F) - \frac{1}{2}E((M + V'y)^T\Sigma^{-1}N(M + V'y)) \right\} \\ &= \text{constant} + \sum_s \left\{ E[y^T V'^T\Sigma^{-1}(F - NM)] - \frac{1}{2}E(y^T V'^T\Sigma^{-1}NV'y) \right\} \\ &= \text{constant} + \sum_s \left\{ \text{tr}[V'^T\Sigma^{-1}(F - NM)(Ey)^T] - \frac{1}{2}\text{tr}[V'^T\Sigma^{-1}NV'E(yy^T)] \right\} \end{aligned} \tag{44}$$

Set the gradient w.r.t. *V'* to zero:

$$\nabla_{V'}Q(V,V') = \sum_s \left[\Sigma^{-1}(F - NM)(Ey)^T - \Sigma^{-1}NV'E(yy^T) \right] = 0 \tag{45}$$

we can find the optimal eigenvoice matrix to be the solution of the following equation system:

$$\sum_s N\hat{V}E(yy^T) = \sum_s (F - NM)(Ey)^T \tag{46}$$

This equation system can be solved row by row.

The entire EM algorithm for training *V* starts by initializing *V* randomly, and going through the above iteration for a number of times. The procedure is summarized below, introducing some new symbols for conciseness:

- For each speaker *s*:
  - Calculate the Baum-Welch statistics $N(s)$ and $F(s)$ against the UBM;
  - Calculate the "centered" first-order Baum-Welch statistics $\tilde{F}(s) = F(s) - N(s)M$;
- Initialize *V* randomly.
- Repeat for a fixed number of times or until convergence:
  - For each speaker *s*:
    - Calculate the matrix $G(s) = (I + V^T\Sigma^{-1}N(s)V)^{-1}$;
    - Calculate the statistics about the posterior distribution of *y*:
      - $E_1(s) = Ey = G(s)V^T\Sigma^{-1}\tilde{F}(s)$; (Eq. (42))
      - $E_2(s) = E(yy^T) = E_1(s)E_1(s)^T + G(s)$; (Eq. (43))

- Solve the equation system $\sum_s N(s)\hat{V}E_2(s) = \sum_s \tilde{F}(s)E_1(s)^T$; (Eq. (46))
- Replace V with $\hat{V}$.

(2) Training the *U* matrix

The procedure for training V is similar to that of training U. The differences include:

- In the main loop, things are calculated for each utterance instead of for each speaker;
- The speaker factors $y(s)$ is estimated for each speaker before the main loop, and subtracted from the centered first-order Baum-Welch statistics of each utterance.

The entire procedure is as below.

- For each speaker *s*:
  - Calculate the Baum-Welch statistics $N(s)$ and $F(s)$ against the UBM;
  - Estimate the speaker factors $y(s) = (I + V^T\Sigma^{-1}N(s)V)^{-1}V^T\Sigma^{-1}(F(s) - N(s)M)$; (Eq. (31))
- For each utterance *u* (suppose it's spoken by *s*):
  - Calculate the Baum-Welch statistics $N(u)$ and $F(u)$ against the UBM;
  - Calculate the "centered" first-order Baum-Welch statistics $\tilde{F}(u) = F(u) - N(u)[M + Vy(s)]$;
- Initialize U randomly.
- Repeat for a fixed number of times or until convergence:
  - For each speaker *u*:
    - Calculate the matrix $G(u) = (I + U^T\Sigma^{-1}N(u)U)^{-1}$;
    - Calculate the statistics about the posterior distribution of *x*:
      - $E_1(u) = Ex = G(u)U^T\Sigma^{-1}\tilde{F}(u)$;
      - $E_2(u) = E(xx^T) = E_1(u)E_1(u)^T + G(u)$;
  - Solve the equation system $\sum_u N(u)\hat{U}E_2(u) = \sum_u \tilde{F}(u)E_1(u)^T$;
  - Replace U with $\hat{U}$.

(3) Training the *D* matrix

- For each speaker *s*:
  - Calculate the Baum-Welch statistics $N(s)$ and $F(s)$ against the UBM;
  - Estimate the speaker factors $y(s) = (I + V^T\Sigma^{-1}N(s)V)^{-1}V^T\Sigma^{-1}(F(s) - N(s)M)$;
- For each utterance *u* (suppose it's spoken by *s*):
  - Calculate the Baum-Welch statistics $N(u)$ and $F(u)$ against the UBM;
  - Estimate the channel factors $x(u) = (I + U^T\Sigma^{-1}N(u)U)^{-1}U^T\Sigma^{-1}[F(u) - N(u)(M + Vy(s))]$;
  - Calculate the "centered" first-order Baum-Welch statistics $\tilde{F}(u) = F(u) - N(u)[M + Vy(s) + Ux(u)]$;
- For each speaker *s*:
  - Calculate the "centered" first-order Baum-Welch statistics $\tilde{F}(s)$ by summing up

the $\tilde{F}(u)$ of all utterances *u* spoken by *s*.

- Initialize *D* to a random diagonal matrix.
- Repeat for a fixed number of times or until convergence:
  - For each speaker *s*:
    - Calculate the matrix $G(s) = (I + D^2\Sigma^{-1}N(s))^{-1}$;
    - Calculate the statistics about the posterior distribution of *z*:
      - $E_1(s) = Ez = G(s)D\Sigma^{-1}\tilde{F}(s)$;
      - $E_2(s) = E(zz^T) = E_1(s)E_1(s)^T + G(s)$; *
  - Solve the equation system $\sum_s N(s)\hat{D}E_2(s) = \sum_s \tilde{F}(s)E_1(s)^T$; *
  - Replace *D* with $\hat{D}$.

Note that, because *D* is a diagonal matrix, for the equations marked with an asterisk, we only need to care about the diagonal elements.

3.3.2 Variants of JFA and Our Choices

(1) Order of training the *V*, *U*, *D* matrices. [14] doesn't state it very clearly but seems to imply that they trained the matrices in the order of *V*-*D*-*U*. However, in most code we have found (including the Matlab demo [15] and Alize), the matrices were trained in the order of *V*-*U*-*D*. Our tutorial above was also written in the order of *VU*-*D*. The Janus code we had (see Section 4.3.1) seemed to support both orders.

The two different training orders affect the data for training *D*. In the *V*-*D*-*U* order, *D* is trained before the eigenchannels, therefore there must be multiple utterances for the same speaker to average out the channel effect. On the other hand, in the *V*-*U*-*D* order, the channel effect in the utterances for training *D* can be removed since *U* has been trained already, therefore it is possible to use only one utterance for each speaker. However, theoretically the *V*-*D*-*U* order makes more sense, since it first trains everything related to speaker variability, and then trains the rest related to channel variability.

We adopted the *V*-*D*-*U* order in our latest experiments.

(2) Initialization of the *V*, *U*, *D* matrices. This is not carefully stated in most literature; usually they just say "random initialization" (e.g. [14]). However, we find that the *order of magnitude* of the initial value is important. We find that after the first iteration, the directions of the columns would have almost converged, and in the subsequent iterations it is mostly the magnitude that gets adjusted. When the initial values are two small, it would take more iterations for the magnitude to converge; when the initial values are too large, the magnitude would decrease at a very slow speed and wouldn't converge even in hundreds of iterations. We came up with the following empirical scheme for initializing these matrices:

Let $L = CF$ be the length of the supervector, $\overline{\Sigma^{-1}}$ be the mean of the inverse of the elements in Σ (the block diagonal matrix of the component covariance matrices), and $\overline{N}$ be the average zeroth-order Baum-Welch statistics (number of points) per component and per speaker. Then, initialize the elements of *V* and *U* to obey the normal distribution

$\mathsf{N}(0,\ 1/L\overline{\Sigma^{-1}}\overline{N})$, and the diagonal elements of D to be the absolute value of random variables that obey the normal distribution $\mathsf{N}(0,\ 1/\overline{\Sigma^{-1}}\overline{N})$.

This initialization scheme has been found to work better than initializing everything to be uniformly distributed within the interval [0,1] (as implemented in Janus and Alize).

(3) Methods and number of iterations in training the *V*, *U*, *D* matrices. The training procedure of the matrices is an iterative procedure, where in each iteration an objective function is increased. For example, in the training of the V matrix, the objective function is as follows:

$$\begin{aligned} L(V) &= \sum_s \log P(X(s)\,|\,V) \\ &= \sum_s \log \int_{y(s)} P(y(s)) \cdot P(X(s)\,|\,y(s), V)\ \mathrm{d}y(s) \end{aligned} \tag{47}$$

Regarding $y(s)$ as a latent variable, we built an auxiliary function

$$Q(V, V') = \text{constant} + \sum_s \int_{y(s)} P(y(s)\,|\,X(s), V) \log P(X(s), y(s)\,|\,V')\ \mathrm{d}y(s) \tag{48}$$

which is maximized in each iteration with a series of updating equations. This procedure is called "maximum likelihood (ML) estimation", because it increases the likelihood function $L(V)$.

Beside the most common ML estimation, in [14], another updating procedure called "minimum divergence (MD) estimation" is used. Despite the name, this procedure also increases the likelihood function $L(V)$ in each iteration. It is also an EM algorithm but chooses a different latent variable than $y(s)$, and its name comes from the auxiliary function which has the form of the KL-divergence between two Gaussian distributions.

The updating equation of MD estimation is:

$$V \leftarrow VJ, \qquad \text{where } JJ^T = \frac{1}{S}\sum_s E_2(s) \tag{49}$$

In the above equation, S is the total number of speakers, and $E_2(s) = E[y(s)y(s)^T]$ can be calculated according to Eq. (43). J is a lower-triangle matrix called the "Cholesky decomposition" of the right hand side. (Some formulas in [14] include another term; it arises because [14] also trains the M in the JFA model, but we keep M fixed to the UBM mean.) A complete derivation of this updating equation can be found in B.

Since MD estimation is only a right-multiplication onto the V matrix, it does not change the subspace spanned by the columns of V (i.e. the channel space), but only performs a linear combination upon the eigenvoice vectors. [14] describes the advantage of MD estimation as "getting good estimates of the eigenvalues corresponding to the eigenvoices".

Most of the code we have seen uses only several ML iterations, but [14] uses 7 ML iterations plus 1 MD iteration. In our correspondence with P. Kenny (first author of [14]), Kenny claims it is necessary to do the MD iteration in order for the magnitudes of the matrix elements to be interpretable. In reality, we found that the MD iteration alters the magnitudes of the eigenvoices / eigenchannels significantly. After the ML iterations, the lengths of the

eigenvoices / eigenchannels are almost equal; but after the MD iteration, these lengths form the shape an exponential function. However, MD iterations did not contribute significantly to the objective function nor to the final performance; also, we found that some rows of V and U tend to diverge in MD iterations.

In our latest experiments we used only 3 ML iterations. We didn't use MD iterations so that we could compare our own implementation with Janus and Alize; we did only 3 iterations because we had a better initialization.

(4) Estimating *x*, *y*, *z* in speaker enrollment. In Section 3.3.1.3, we showed how to estimate these vectors in two steps (Eqs. (32) and (33)). In fact, it is also possible to estimate them jointly. The solution, which involves more complicated math, is given in C. The two-step estimation is an approximation of the exact joint estimation, but the deviation is small enough.

The references [16] and [18] introduce a "Gauss-Seidel-like iterative algorithm", which also yields an approximate solution. It has the advantage of being easy to implement. However, since we have already implemented joint estimation, we stuck to it in our experiments.

3.3.3 Scoring Methods Special to JFA

In [13], a series of scoring methods special to JFA are compared. With the exception of full frame-by-frame scoring, all scoring methods are approximate and are based on the Baum-Welch statistics. The score of a trial utterance against a target speaker is expressed as a function of the Baum-Welch statistics N, F of the trial utterance (calculated against the UBM), the supervector m of the target speaker model, and the concatenated covariance matrix Σ. Usually this score is adjusted by subtracting the score of the utterance against the UBM. The adjusted score is called likelihood ratio (LLR), and will also contain the UBM supervector M.

(1) "nox" – no consideration of channel factors

A preliminary approximation is to replace the exact frame-average log-likelihood in Eq. (7) with the EM auxiliary function, which simplifies to:

$$\text{score}_{\text{nox}}(X, m) = \frac{1}{n}[\text{constant} + m^T \Sigma^{-1}(F - \frac{1}{2} Nm)] \tag{50}$$

where X stands for the trial utterance, and n is the length (number of feature vectors) of X. The "constant" term does not depend on m. The LLR will be

$$\text{LLR}_{\text{nox}}(X, m) = \frac{1}{n}[m^T \Sigma^{-1}(F - \frac{1}{2} Nm) - M^T \Sigma^{-1}(F - \frac{1}{2} NM)] \tag{51}$$

(2) "intx" – integrating out the channel factors

Recall that when training the speaker models, the channel factors x are discarded. This means the target speaker supervectors m (as well as the UBM supervector M) doesn't contain any channel factors. Scoring a trial utterance against m effectively ignores the channel factors x that were at work in producing X, giving rise to mismatch.

Because nothing is known about the channel factors x of X, a wise way to deal with them is to integrate x out, using its standard normal prior distribution:

$$\text{score}_{\text{intx}}(X, m) = \frac{1}{n} \int_x [\text{constant} + (m + Ux)^T \Sigma^{-1}(F - \frac{1}{2} N(m + Ux))]\, p(x; 0, 1)\, \text{d}x \tag{52}$$

This works out to

$$\text{score}_{\text{intx}}(X,m)=\frac{1}{n}[\text{constant}+(F-Nm)^T\Sigma^{-1}UGU^T\Sigma^{-1}(F-Nm)+m^T\Sigma^{-1}(F-\frac{1}{2}Nm)] \tag{53}$$

where $G=(I+U^T\Sigma^{-1}NU)^{-1}$. The LLR is just

$$\text{LLR}_{\text{intx}}(X,m)=\text{score}_{\text{intx}}(X,m)-\text{score}_{\text{intx}}(X,M) \tag{54}$$

(3) "pointx" – point estimate the channel factors

Instead of integrating out the channel factors *x*, we can also find a point estimate of *x* that maximizes the posterior probability of *X*, i.e. maximizes the integrand of Eq. (52). The solution is given by

$$\hat{x}_m=GU^T\Sigma^{-1}(F-Nm) \tag{55}$$

Therefore the score function is

$$\text{score}_{\text{pointx}}(X,m)=\frac{1}{n}[\text{constant}+(m+U\hat{x}_m^T)\Sigma^{-1}(F-\frac{1}{2}N(m+U\hat{x}_m)] \tag{56}$$

To calculate $\text{score}_{\text{pointx}}(X,M)$, another point estimate of *x* is needed:

$$\hat{x}_M=GU^T\Sigma^{-1}(F-NM) \tag{57}$$

$$\text{score}_{\text{pointx}}(X,M)=\frac{1}{n}[\text{constant}+(M+U\hat{x}_M^T)\Sigma^{-1}(F-\frac{1}{2}N(M+U\hat{x}_M)] \tag{58}$$

The LLR is just

$$\text{LLR}_{\text{pointx}}(X,m)=\text{score}_{\text{pointx}}(X,m)-\text{score}_{\text{pointx}}(X,M) \tag{59}$$

(4) "ubmx" – UBM point estimate the channel factors

An approximation is made by replacing $\hat{x}_m$ with $\hat{x}_M$.

$$\text{score}_{\text{ubmx}}(X,m)=\frac{1}{n}[\text{constant}+(m+U\hat{x}_M^T)\Sigma^{-1}(F-\frac{1}{2}N(m+U\hat{x}_M)] \tag{60}$$

$$\text{score}_{\text{ubmx}}(X,M)=\frac{1}{n}[\text{constant}+(M+U\hat{x}_M^T)\Sigma^{-1}(F-\frac{1}{2}N(M+U\hat{x}_M)] \tag{61}$$

$$\text{LLR}_{\text{pointx}}(X,m)=\frac{1}{n}[(m-M)^T\Sigma^{-1}(F-NM-NU\hat{x}_M)-\frac{1}{2}(m-M)^T\Sigma^{-1}N(m-M)] \tag{62}$$

(5) "linear" – linear approximation of (4)

By assuming that $m-M$ is small, the quadratic term in (62) is dropped, resulting in the following linear LLR function:

$$\text{LLR}_{\text{linear}}(X,m)=\frac{1}{n}(m-M)^T\Sigma^{-1}(F-NM-NU\hat{x}_M) \tag{63}$$

The paper [13] compares full frame-by-frame scoring and the scoring methods (2~5) here as follows:

- In terms of accuracy:
  - Without score normalization, full frame-by-frame is best; intx, pointx and linear

come close; ubmx is by far the worst.
    - With ZT-norm, full frame-by-frame, intx, pointx and linear are all similar, with pointx and linear reaching the best under some conditions; ubmx is somewhat worse.
- In terms of efficiency: full frame-by-frame is very slow; pointx is a bit slow; the other three are fast.
- Considering both aspects, linear scoring seems to be a best choice.

The "nox" scoring method is applicable to GMM modeling without JFA. The linear scoring is also applicable, if $U = 0$ is substituted into Eq. (63). The other three scoring methods are not applicable to GMM modeling without JFA.

3.4 Score Normalization

The score of an utterance against a speaker model measures their similarity. However, a larger score doesn't always mean they're more similar. For example, a speaker model may tend to yield higher scores than other models for imposter utterances. In this case, even if an utterance scores high against this model, it doesn't necessarily mean that this is the most likely model. Also, an utterance may tend to score higher than other utterances against models that do not represent the speaker of this utterance. In this case, the utterance has to score even higher against a speaker model to be considered a match. In a word, what matters is not the score itself, but rather how much it stands out from the scores of imposter trials. Unfortunately the distribution of scores of imposter trials may be different for each model and each utterance. This is problematic especially for open-set SID, because we need to set a global threshold θ for all trials. Score normalization is the technique that normalizes the distribution of scores of imposter trials, so that a fairer comparison of models could be made in closed-set SID, and a global threshold could be set in open-set SID.

Two common normalization techniques are zero normalization (Z-norm) and test normalization (T-norm). Z-norm normalizes the scores by model:

$$\text{score}_{\text{Znorm}}(X, m) = \frac{\text{score}(X, m) - \mu_m}{\sigma_m} \tag{64}$$

where μ_m and σ_m are the mean and variance of the distribution of scores of imposter trials against the model *m*. The imposter utterances are taken from the development data, and the parameters μ_m and σ_m can be pre-computed in the training stage. On the other hand, T-norm normalizes the scores by utterance:

$$\text{score}_{\text{Tnorm}}(X, m) = \frac{\text{score}(X, m) - \mu_X}{\sigma_X} \tag{65}$$

where μ_X and σ_X are the mean and variance of the distribution of scores of *X* against a set of cohort models. The cohort models are trained from another set of utterances taken from the development data. Since the parameters μ_X and σ_X rely on the test utterance, they must be

estimated in the testing stage.

The two normalization techniques can be applied one after the other, and the compound normalization techniques are called ZT-norm and TZ-norm. In ZT-norm, we first normalize the scores (of both target models and cohort models) with Z-norm, and then normalize the resulting scores with T-norm. In TZ-norm, we first normalize the scores (of both the trial utterances and the development imposter utterances) with T-norm, and then normalize the resulting scores with Z-norm. In order to ensure that the scores used as references are indeed from imposter trials, the cohort set and the development imposter set must not overlap.

We studied the effect of score normalization with open-set SID experiments on the ROSSI database. The numbers are given in Section 4.2. We found that T-norm improved the open-set performance (while not affecting closed-set performance), while the other normalizations decreased the performance most of the time.

4. RESULTS AND DISCUSSION

4.1 Closed-Set SID Experiments

We compare the performance of the most of the acoustic features and pre-/post-processing techniques by the closed-set accuracy on the ROSSI database. The numbers are shown in Table 2.

The naming convention is as follows: The core part of the feature name is the feature type and its dimensionality, for example, "MFCC21" means 21-dimensional MFCC feature. If the dimensionality is doubled (e.g. MFCC42), then it means delta features are appended. Pre-/post-processing techniques are attached to the feature name as prefixes or suffixes. "PRE" stands for pre-emphasis, "LP" stands for linear prediction, "CMN" stands for cepstral mean normalization, and "GAUSS" stands for short-time Gaussianization.

From the table we can see that the features and pre-/post-processing techniques that improve the closed-set accuracy include:

- Multitaper MFCC (especially using 8 multipeak tapers);
- DSCC;
- Pre-emphasis and delta features;
- Short-time Gaussianization.

Table 2. Performance of Various Acoustic Features and Pre-/Post-Processing Techniques, Compared by Closed-Set Accuracy on the ROSSI Database

Feature	Set ID								
	1	2	3	4	5	6	7	8	Average
MFCC21_CMN (Baseline)	89	72	63	43	41	40	65	61	59.250
MFCC42_CMN	86	75	66	43	44	48	65	62	61.125
PRE_MFCC21_CMN	91	75	62	46	41	38	67	60	60.000
PRE_MFCC42_CMN	92	77	65	48	48	46	70	61	63.375

WMVDR21_CMN		88	74	64	43	40	40	65	62	59.500
SCF20		63	58	44	17	17	15	44	43	37.625
SCF20_PCA		65	64	50	20	21	15	47	46	41.000
MFCC21_CMN + SCF20		75	68	56	30	24	26	57	50	48.250
MFCC21_CMN + SCF20_PCA		83	68	60	33	26	23	59	54	50.750
MFCC21_CMN + DSCC20_GAUSS100_CMN (Compared with MFCC42_CMN)		82	78	69	48	46	47	66	64	62.500
Multitaper MFCC	**Baseline (1 Hamming)** – our implementation	89	72	63	43	41	40	65	61	59.250
	Baseline (1 Hamming) – Kinnunen's implementation	88	71	64	50	40	40	63	61	59.625
	4 Thompson	86	70	66	49	44	38	67	63	60.375
	8 Multipeak	90	75	68	50	48	39	64	62	62.000
	8 SWCE	91	75	65	46	48	34	67	62	61.000
LP_MFCC21_CMN (Param = LP order)	**1**	86	72	63	43	40	38	65	61	58.500
	2	86	76	64	47	41	39	65	59	59.625
	5	85	75	65	41	31	35	59	62	56.625
	10	83	73	68	37	33	30	59	57	55.000
	15	81	72	65	37	34	31	60	58	54.750
	20	81	71	64	38	33	30	59	57	54.125
MFCC21_GAUSS (Param = window length in frames)	**100**	86	74	64	51	42	42	61	64	60.500
	200	88	76	65	55	43	43	62	65	62.125
	300	85	76	66	51	45	43	64	64	61.750
	400	86	75	66	52	44	46	66	62	62.125
	500	85	76	67	54	45	44	68	64	62.875
PRE_MFCC42_GAUSS (Param = window length in frames) (Compared with PRE_MFCC21_CMN + DELTA)	**100**	89	75	72	54	43	49	66	64	64.000
	200	93	75	72	55	45	50	66	67	65.375
	300	91	76	70	58	45	48	67	65	65.000
	400	93	78	68	56	46	50	68	65	65.500
	500	92	76	68	58	45	47	66	64	64.500

For the prosodic features HSCC and FFV, and their fusion with the baseline MFCC, we evaluated their performance on the four scenarios on the MIXER5 database, also using the closed-set accuracy as the criterion. The numbers are shown in Table 3, and the change of the performance with the fusion weights is visualized in Figure 6. The improvement of the fusion over baseline MFCC is minor, and the fusion weights are hard to explain: in three of the four scenarios, FFV, which performs worst alone, has (one of) the highest weights.

Table 3. Performance of HSCC, FFV, and Their Fusion with MFCC, Compared by Closed-Set Accuracy on the MIXER5 Database

	VB-YB	VL-YL	VB-YL	VL-YB
MFCC21	70.43	74.71	59.29	55.43
HSCC20	38.43	44.14	23.00	23.00
FFV7	32.29	36.57	10.29	13.71
Best fusion	74.71	79.86	61.71	56.86
Best fusion weights	0.39 + 0.03 + 0.58	0.35 + 0.05 + 0.60	0.47 + 0.06 + 0.47	0.87 + 0.04 + 0.09

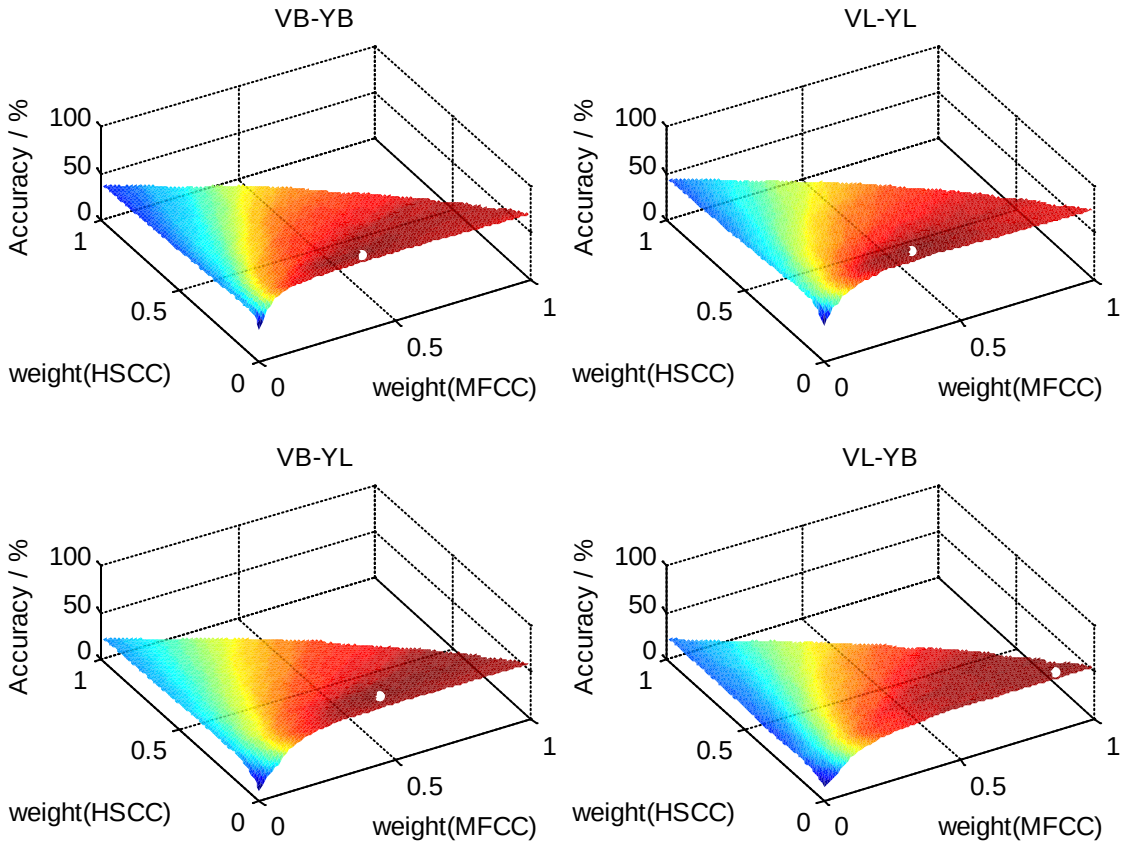

Figure 6. Fusion of MFCC, HSCC and FFV: How the Closed-Set Accuracy Changes with the Fusion Weights

4.2 Open-Set SID Experiments

We conducted extensive open-set SID experiments on the ROSSI database. The variables we considered include:

- **Acoustic features**: six types of features that investigate the effect of pre-emphasis, delta feature, short-time Gaussianization, and multitapers. For short-time Gaussianization, we chose a window length of 400 frames which has been shown to

perform best for closed-set SID. For multitapers, we used 8 Multipeak tapers.

- **Speaker modeling method and kernel function**: we compared GMM speaker modeling and SVM speaker modeling; for the latter, we considered all the six kernel functions.
- **Score normalization**: No normalization, Z-norm, T-norm, ZT-norm and TZ-norm.

For each combination of variables, we measured the 4 criteria for each of the 8 sets of the ROSSI database. This yielded a total of $6 \times 7 \times 5 \times 8 \times 4 = 6720$ numbers, which were too many to analyze. Therefore we aggregated the numbers in the following ways:

- The numbers across the 8 sets are averaged;
- For the six kernel functions, only the best is reported along with the name of that kernel.

Now there are $6 \times 2 \times 5 \times 4 = 240$ numbers, which is still quite a lot but can be analyzed. These numbers are listed in Figure 7.

Figure 7 is made up of five rows and two columns, where each row stands for a type of score normalization and the two columns stand for the two scoring methods. Each small table has five rows corresponding to features and four columns corresponding to the evaluation criteria. In the SVM modeling part, the name of the best kernel is shown alongside the numbers. The greener the background is, the larger the number. The maximum number in each column of the small tables is highlighted in red.

The following conclusions can be obtained concerning the variables we're interested in:

- **Scoring method**: SVM speaker modeling is significantly better than GMM modeling.
- **Score normalization**: T-norm improves the open-set performance (while not affecting closed-set performance), while the other normalizations decrease the performance most of the time.
- **Acoustic feature**: The best feature for closed-set SID is PRE_PEAK8_MFCC40_GAUSS400 (with every technique included). The best feature for open-set SID, if we focus on the T-norm, is PRE_MFCC42_GAUSS400. Multitapers do not seem to work well for open-set SID. For other score normalizations the conclusion may be a bit different, but there's no doubt that the pre-emphasis and delta features should be included.
- **Kernel function**: The numbers favor the WBHATT kernel for closed-set SID, and the BHATT or GUMI kernel for open-set SID. Experiments that compare the performances of different kernel functions show that the PLAIN and KL kernels also achieve a similar performance with negligible differences, but the L2 kernel performs much worse and can be eliminated.

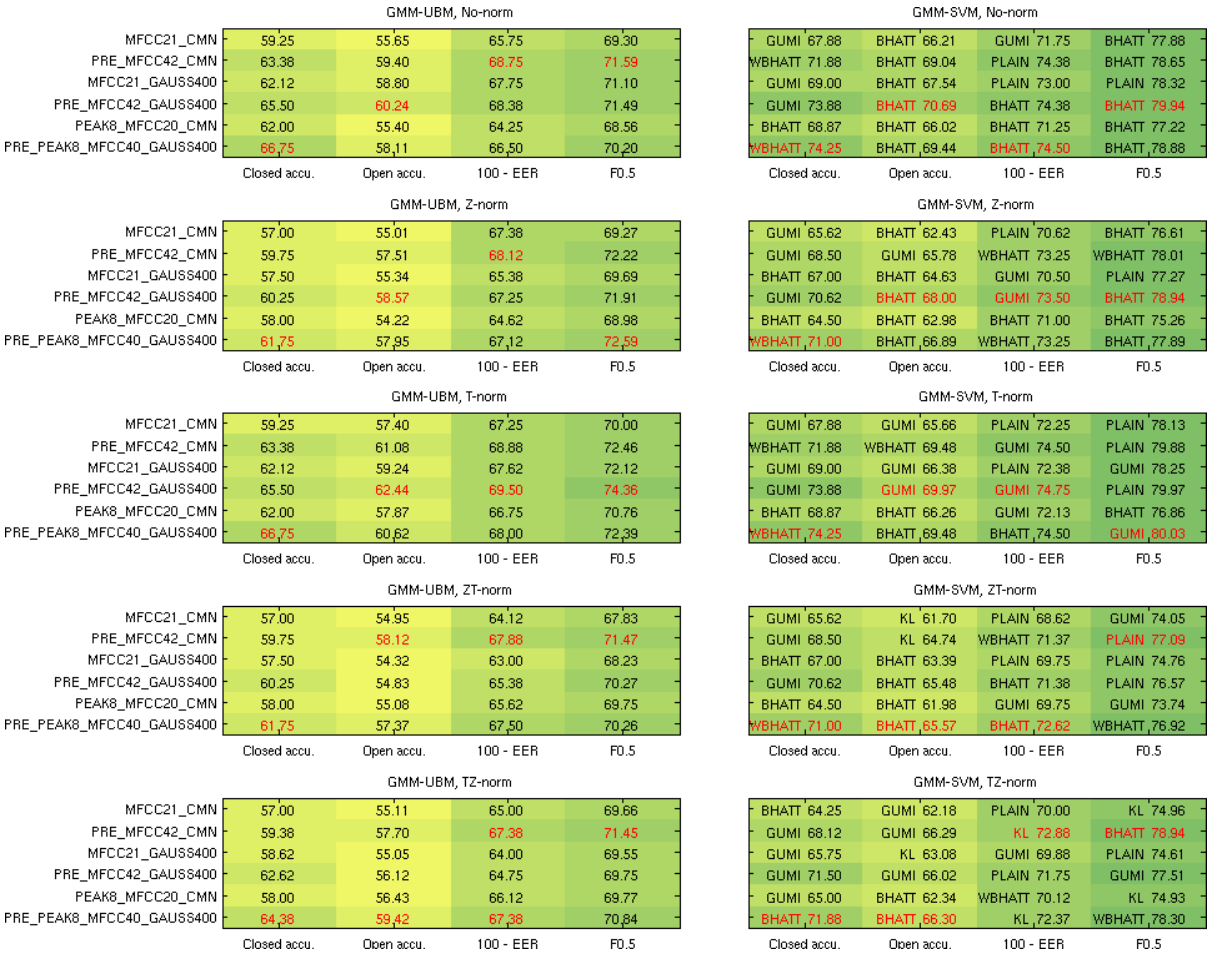

Figure 7. Open-Set Evaluation Results on the ROSSI Database

4.3 Experiments with JFA

4.3.1 Toolkits Used in Our Experiments of JFA

We implemented everything except frame-by-frame scoring in Matlab. Because our results didn't turn out good for a long time, we also used two toolkits for comparison: Janus and Alize. We compared the part of the speaker recognition pipeline after calculating the Baum-Welch statistics, including the initialization and training of the *V*, *U*, *D* matrices, speaker enrollment, and scoring.

Janus refers to the JFA functionality built into the "Janus" speech recognition software of our lab, written in C++. It initializes the JFA matrices at the order of magnitude of 1, but also supports custom initialization. The training order of the matrices seems to be flexible. The enrollment and scoring parts are too slow to be used.

Alize [19] is an open source platform for biometric authentication developed by the University of Avignon, including the functionality of JFA. It is also written in C++. Like Janus, Alize initializes JFA matrices at the order of magnitude of 1 by default and supports custom initialization. It must train the JFA matrices in the order of *V*-*U*-*D*. In speaker enrollment, Alize first estimates the vectors *y* and *x* jointly, then estimates the *z* vector. The

scoring methods offered by Alize include frame-by-frame scoring (either full or fast) and linear scoring.

4.3.2 JFA Experiment Setup and Results

In our experiments, we used our own initialization of the JFA matrices (which has been found to be superior to the default initialization of Janus and Alize). We compared the training of the JFA matrices of our own code and the two toolkits (*V-D-U* with Matlab and Janus, *V--U-D* with Alize). For speaker enrollment, we used our own code of joint estimation of *y*, *x* and *z*. The two-step estimation of Alize doesn't exhibit much difference from joint estimation. For scoring, we used Alize's fast frame-by-frame scoring (choosing 5 top components) and our implementation of the five scoring methods in Section 3.3.3. It has been verified that Alize's implementation of linear scoring produces exactly the same results as our implementation. We also implemented SVM modeling and scoring.

Because we didn't get good results, and because the results seem to be sensitive to many variables, we used only the PLAIN kernel for SVM modeling and scoring, and didn't apply any score normalization.

Below are our latest results (EER) obtained on the sixth subset (telephone train, telephone test) of the male trials of the NIST 2008 evaluation. The acoustic feature used is RM40.

Table 4. Comparison of EER of various training methods of JFA matrices and scoring methods on the sixth subset of NIST 2008 male trials

	Relevance MAP	JFA with Matlab	JFA with Janus	JFA with Alize
Fast frame-by-frame	14.76	15.66	14.32	15.90
nox	14.12	15.38	15.10	15.21
intx	N/A	15.34	11.85	15.15
pointx		15.34	11.86	15.16
ubmx		15.33	16.70	15.22
linear	12.93	16.65	12.47	16.70
SVM (PLAIN kernel)	9.77	11.78	11.78	11.72

It appears that among the three implementations of JFA, only Janus shows an improvement over the relevance MAP baseline (except the case with SVM). Also, the trend of the performance of the intx, pointx, ubmx and linear scoring methods agrees with those in [13]. However, when we investigated what was the difference of the Janus implementation from the others, we found that the *U* matrix wasn't correctly trained by Janus: 98 out of the 100 columns were almost identical. We tried running the enrollment and scoring with only some of the matrices *V*, *U*, *D*, and found that it was exactly the "wrong" *U* matrix that produced the low error rates. We tried training the *U* (or *D*) matrix directly after training the *V*

matrix with the three implementations, and found that Janus produced different results from the other two. The *D* matrix trained by Janus even contained negative values despite the all-positive initialization. This seems to be evidence that the Janus implementation is probably wrong, and the good numbers we obtained couldn't be explained. Also, when SVM modeling and scoring are used, we never found a configuration that beat the relevance MAP baseline. We suspect that the following two factors might relate to the no success of JFA:

1. The database used, and the division of the data. The NIST database is huge, and its organization is complicated. It might take pages to exactly specify what data one used for what part of the experiments, and sometimes one may even have to provide the file lists. As a result, we haven't been able to replicate the data division of any literature, and it is possible that we have run all our experiments on a not so reasonable division of the data.

2. Lack of a working configuration of JFA. Although we had access to the source code of two toolkits, the source code only contained functions that perform the individual units of functionality in JFA, and we had to configure the pipeline (e.g. order of training the matrices) on ourselves. In doing so we naturally tended to imitate our own configuration, and if our own configuration had been wrong, the toolkits wouldn't have helped us discover our mistake. Even though the toolkits can be useful for debugging the implementation, we might still have missed the "correct" configuration.

5. CONCLUSION

During the performance period of this project, we have worked on all the modules of speaker recognition systems. At the front end, we have studied a variety of acoustic features and pre-/post-processing techniques, and have come up with a PPMD feature that combines the benefits of multitaper MFCC, DSCC, pre-emphasis, and short-time feature Gaussianization. At the speaker modeling and scoring stages, we have also investigated SVM and JFA. We have demonstrated that compared to GMM modeling, SVM modeling and scoring are not only better and also faster. We have also shown that T-norm of the scores improves open-set speaker identification performance on the ROSSI database.

6. REFERENCES

[1] H. Hermansky and N. Morgan, "Rasta processing of speech," *IEEE Transactions on Speech and Audio Processing*, vol.2, no. 4, pp. 578-589, Oct 1994.

[2] S. O. Sadjadi and J. H. L. Hansen, "Assessment of single-channel speech enhancement techniques for speaker identification under mismatched conditions," *ISCA Interspeech*, pp. 2138-2141, 2010.

[3] Q. Jin, R. Li, Q. Yang, K. Laskowski, and T. Schultz, "Speaker identification with distant microphone speech," *IEEE ICASSP*, pp. 4518-4521, 2010.

[4] T. Kinnunen, *et al*., "What else is new than the Hamming window? Robust MFCCs for speaker recognition via multitapering," *InterSpeech 2010*.

[5] J. M. K. Kua, et al., "Investigation of spectral centroid magnitude and frequency for speaker recognition," Odyssey, pp. 34-39, 2010.

[6] K. Laskowski and Q. Jin, "Modeling prosody for speaker recognition: Why estimating

pitch may be a red herring," *Odyssey*, 2010.
[7] K. Laskowski, J. Edlund, and M. Heldner, "Learning prosodic sequences using the fundamental frequency variation spectrum," *Proc. SPEECH PROSODY*, 2008.
[8] K. Kumar, "A spectro-temporal framework for compensation of reverberation for speech recognition," PhD thesis, Jan 2011, Chapter 8.
[9] A. Kawamura, K. Fujii, Y. Itoh, and Y. Fukui, "A noise reduction method based on linear prediction analysis," *Electronics and Communications in Japan, Part 3*, vol. 86, no. 3, 2003.
[10] B. Xiang, *et al.*, "Short-time Gaussianization for robust speaker verification," *ICASSP*, pp. 681-684, May 2002.
[11] D. Reynolds, T. Quatieri and R. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital Signal Processing*, vol. 10, pp. 19-41, 2000.
[12] P. Kenny, "Joint factor analysis of speaker and session variability: Theory and algorithms," Technical report CRIM-06/08-13 Montreal, CRIM, 2005. [Online] http://www.crim.ca/perso/patrick.kenny/.
[13] O. Glembek, L. Burget, N. Dehak, N. Brümmer, and P. Kenny, "Comparison of scoring methods used in speaker recognition with joint factor analysis," *Proc ICASSP 2009*, Taipei, Taiwan, April 2009.
[14] P. Kenny, P. Ouellet, N. Dehak, V. Gupta, and P. Dumouchel, "A study of inter-speaker variability in speaker verification," *IEEE Trans. on ASLP*, July 2008.
[15] O. Glembek, "Joint factor analysis Matlab demo," [Online] http://speech.fit.vutbr.cz/software/joint-factor-analysis-matlab-demo.
[16] L. Burget *et al.*, "Investigation into variants of joint factor analysis for speaker recognition," *Proc InterSpeech 2009*, pp. 1263-1266, September 2009.
[17] P. Kenny, G. Boulianne, and P. Dumouchel, "Eigenvoice modeling with sparse training data," *IEEE Trans. on Speech and Audio Processing*, vol. 13, no. 3, pp. 345-359, May 2005.
[18] R. Vogt and S. Sridharan, "Explicit modeling of session variability for speaker verification," *Computer Speech and Language*, pp. 17-38, vol. 22, no. 1, January 2008.
[19] "ALIZE wiki,", [Online] http://mistral.univ-avignon.fr/mediawiki/index.php/Main_Page
[20] R. Auckenthaler, M. Carey and H. Lloyd-Thomas, "Score normalization for text-independent speaker verification systems," *Digital Signal Processing*, vol. 10, pp. 42-54, 2000.

A. DERIVATION OF GMM KERNELS

In this appendix we give the derivation of the six GMM kernel functions (PLAIN, GUMI, KL, L2, BHATT, WBHATT). The motivation of these kernels can be divided into three categories:

- Defining a "supervector" for each GMM, and using the inner product of the supervectors as the inner product of the GMMs (PLAIN);
- Defining a "distance" measure between GMMs, and deriving an inner product from the distance (GUMI, KL);
- Directly defining an "inner product" on the probability density functions of GMMs (L2, BHATT, WBHATT).

In the following discussion, we only consider the adaptation of GMM means. The covariances and component weights of the adapted GMMs are identical to those of the UBM. We shall denote the UBM by $M = (\mu_i, \Sigma_i, w_i)$, and the adapted GMMs by $M_a = (\mu_i^a, \Sigma_i, w_i)$ and $M_b = (\mu_i^b, \Sigma_i, w_i)$.

A.1 The PLAIN kernel

The first way of defining a GMM kernel is to define a supervector for each GMM. The PLAIN kernel constructs the supervector by concatenating all the component means. The resulting kernel function is:

$$K_{\text{PLAIN}}(M_a, M_b) = \sum_i (\mu_i^a)^T (\mu_i^b) \tag{A-1}$$

This kernel is simple, but one might criticize it for not taking into account the covariance matrices. For example, it is reasonable to introduce the covariance matrices in a way similar to the Mahalanobis distance:

$$K(M_a, M_b) = \sum_i (\mu_i^a)^T \Sigma_i^{-1} (\mu_i^b) \tag{A-2}$$

As it turns out, this is the GUMI kernel to be discussed next.

A.2 The GUMI kernel [A-1]

The second way of defining a GMM kernel is to derive it from a distance measure. The distance d associated with an inner product $\langle \cdot, \cdot \rangle$ satisfies $d^2(a,b) = \langle a-b, a-b \rangle$. If the squared distance takes on a form in which the difference of some function of the GMM models $f(M_a) - f(M_b)$ occurs twice in a multiplicative relationship, then we can define the replace them with $f(M_a)$ and $f(M_b)$ respectively to define a kernel function. The GUMI

and KL kernels are two examples.

The GUMI (GMM-UBM mean interval) kernel is derived from the Bhattacharyya distance of two probability distributions. The Bhattacharyya distance between two probability distributions $p_a(x)$ and $p_b(x)$ is defined as:

$$D_{\text{Bhatt}}(p_a, p_b) = -\ln \int \sqrt{p_a(x) p_b(x)}\,\mathrm{d}x \tag{A-3}$$

For two Gaussian distribution (not GMMs) $N_a(\mu^a, \Sigma^a)$ and $N_b(\mu^b, \Sigma^b)$, the Bhattacharyya distance is:

$$D_{\text{Bhatt}}(N_a, N_b) = \frac{1}{8}(\mu^a - \mu^b)^T \left(\frac{\Sigma^a + \Sigma^b}{2}\right)^{-1} (\mu^a - \mu^b) \tag{A-4}$$

When the two distributions have identical covariance matrices ($\Sigma^a = \Sigma^b = \Sigma$), the Bhattacharyya distance reduces to:

$$D_{\text{Bhatt}}(N_a, N_b) = \frac{1}{8}(\mu^a - \mu^b)^T \Sigma^{-1} (\mu^a - \mu^b) \tag{A-5}$$

For GMMs, however, it is hard to calculate the Bhattacharyya distance. Nevertheless, the effect of GMM adaptation manifests itself in a shift of the means of the individual components, so it still makes sense if we measure the distance between corresponding components. A GUMI kernel, therefore, is defined for two Gaussian distributions by modifying Eq. (A-5) and dropping the constant:

$$K_{\text{GUMI}}(N_a, N_b) = (\mu^a)^T \Sigma^{-1} (\mu^b) \tag{A-6}$$

And the GUMI kernel for GMMs is then defined as the sum of Eq. (A-6) over all components:

$$K_{\text{GUMI}}(M_a, M_b) = \sum_i (\mu_i^a)^T \Sigma_i^{-1} (\mu_i^b) \tag{A-7}$$

(In the original literature the GUMI kernel is defined as

$$K_{\text{GUMI}}(M_a, M_b) = \sum_i (\mu_i^a - \mu_i)^T \Sigma_i^{-1} (\mu_i^b - \mu_i) \tag{A-8}$$

But since the SVM is shift-invariant, the subtraction of the UBM means can be dropped.)

A.3 The KL kernel [A-1][A-2]

The KL kernel is based on the KL divergence of two probability distributions. The KL divergence from a probability distribution $p_a(x)$ to another probability distribution $p_b(x)$ is defined as:

$$D_{\text{KL}}(p_a \,||\, p_b) = \int p_a(x) \log \frac{p_a(x)}{p_b(x)}\mathrm{d}x \tag{A-9}$$

Note that this is an asymmetric function. For Gaussian distributions $N_a(\mu^a, \Sigma^a)$ and $N_b(\mu^b, \Sigma^b)$, the KL divergence is:

$$D_{\mathrm{KL}}(N_a \,||\, N_b) = \frac{1}{2}(\mu^a - \mu^b)^T (\Sigma^b)^{-1}(\mu^a - \mu^b) \tag{A-10}$$

Note the asymmetry: the KL divergence depends on the covariance Σ^b but not Σ^a. However, when the two distributions have identical covariances, the KL divergence is reduced to:

$$D_{\mathrm{KL}}(N_a \,||\, N_b) = \frac{1}{2}(\mu^a - \mu^b)^T \Sigma^{-1}(\mu^a - \mu^b) \tag{A-11}$$

which is essentially the same as the Bhattacharyya distance (Eq. (A-5)).

From this point on we may derive the same kernel function as the GUMI kernel. But the authors of [A-2] didn't want to jump from the KL divergence of GMMs to the sum of KL divergences of individual components directly. Instead, they worked out an upper bound of the KL divergence of GMMs using the log-sum inequality:

$$\begin{aligned} D_{\mathrm{KL}}(M_a \,||\, M_b) &\le \sum\nolimits_i w_i D_{KL}(M_i^a \,||\, M_i^b) \\ &= \sum\nolimits_i w_i (\mu_i^a - \mu_i^b)^T \Sigma_i^{-1} (\mu_i^a - \mu_i^b) \end{aligned} \tag{A-12}$$

where M_i^a and M_i^b stand for the *i*-th component of the two GMMs. And from this they derived the KL kernel function:

$$K_{\mathrm{KL}}(M_a, M_b) = \sum\nolimits_i w_i (\mu_i^a)^T \Sigma_i^{-1} (\mu_i^b) \tag{A-13}$$

It turns out that the sole difference between the GUMI and KL kernel is whether the components are weighted. There is some approximation involved in the derivation of both kernels, so there is no theoretical justification for either of them. The paper [A-1] says that "in various signal selection problems, the Bhattacharyya distance has shown to give better results than the KL divergence," but we still need to do experiments with our own data to see the difference in performance.

A.4 The L2 kernel [A-2]

The third way to define a GMM kernel function is to treat GMMs as continuous probability density functions, and use a general inner product for continuous functions. The L2 kernel makes use of the L2 inner product of real continuous functions:

$$\left\langle f(x), g(x) \right\rangle_{\mathrm{L2}} = \int f(x) g(x) \mathrm{d}x \tag{A-14}$$

For two Gaussian distributions $N_a(\mu^a, \Sigma^a)$ and $N_b(\mu^b, \Sigma^b)$ in a *D*-dimensional space, the inner product is:

$$\begin{aligned} \left\langle N_a, N_b \right\rangle_{\mathrm{L2}} &= N(\mu^a - \mu^b \,|\, 0, \Sigma^a + \Sigma^b) \\ &= \frac{1}{(2\pi)^{D/2} \left|\Sigma^a + \Sigma^b\right|^{1/2}} \exp\left[-\frac{(\mu^a - \mu^b)^T (\Sigma^a + \Sigma^b)^{-1} (\mu^a - \mu^b)}{2} \right] \end{aligned} \tag{A-15}$$

With identical covariance matrices, the inner product is reduced to:

$$\left\langle N_a, N_b \right\rangle_{\mathrm{L2}} = N(\mu^a - \mu^b \mid 0, 2\Sigma)$$
$$= \frac{1}{(4\pi)^{D/2} \left|\Sigma\right|^{1/2}} \exp\left[-\frac{(\mu^a - \mu^b)^T \Sigma^{-1} (\mu^a - \mu^b)}{4} \right] \quad \text{(A-16)}$$

The inner product of two GMMs is:

$$\left\langle M_a, M_b \right\rangle_{\mathrm{L2}} = \sum_i \sum_j w_i w_j \left\langle M_i^a, M_i^b \right\rangle_{\mathrm{L2}} \quad \text{(A-17)}$$

Under the assumption that non-corresponding components of the two GMMs are situated far away so that they do not make significant contribution to the inner product, this is approximated to yield the L2 kernel function (dropping the constant term $(4\pi)^{D/2}$:

$$K_{\mathrm{L2}}(M_a, M_b) = \sum_i w_i^2 \left\langle M_i^a, M_i^b \right\rangle_{\mathrm{L2}}$$
$$= \sum_i \frac{w_i^2}{\left|\Sigma_i\right|^{1/2}} \exp\left[-\frac{(\mu_i^a - \mu_i^b)^T \Sigma_i^{-1} (\mu_i^a - \mu_i^b)}{4} \right] \quad \text{(A-18)}$$

The L2 kernel is somewhat to the radial basis kernel function, because it puts the quadratic form in an exponential function.

A.5 The BHATT and WBHATT kernels

The radial basis kernel function has a nice property: because of the positive semidefinity of the Σ^{-1} matrix, the exponential term is bounded in [0,1]. However, in the L2 kernel, the $\left|\Sigma\right|^{1/2}$ term breaks this bound. Is it possible to eliminate the latter term? The answer is yes – if we use the Bhattacharyya inner product instead of the L2 inner product.

In the definition of the Bhattacharyya distance (Eq. (A-3)), the part inside the logarithm is also a valid inner product for two continuous functions:

$$\left\langle f(x), g(x) \right\rangle_{\mathrm{Bhatt}} = \int \sqrt{f(x) g(x)}\, \mathrm{d}x \quad \text{(A-19)}$$

Because the geometric mean is smaller than or equal to the arithmetic mean, for two probability distributions (whose integral is 1), the Bhattacharyya inner product is bounded in [0,1]. The Bhattacharyya inner product for two Gaussians $N_a(\mu^a, \Sigma)$ and $N_b(\mu^b, \Sigma)$ with identical covariance matrices is:

$$\left\langle N_a, N_b \right\rangle_{\mathrm{Bhatt}} = \exp\left[-\frac{1}{8} (\mu^a - \mu^b)^T \Sigma^{-1} (\mu^a - \mu^b) \right] \quad \text{(A-20)}$$

Similar to the derivation of the L2 kernel, we can derive the weighted Bhattacharyya (WBHATT) kernel for GMMs:

$$K_{\mathrm{WBHATT}}(M_a, M_b) = \sum_i w_i \exp\left[-\frac{(\mu_i^a - \mu_i^b)^T \Sigma_i^{-1} (\mu_i^a - \mu_i^b)}{8} \right] \quad \text{(A-21)}$$

Just like the GUMI kernel, one could argue that the weighting of components is unnecessary,

and thus we have the unweighted Bhattacharyya (BHATT) kernel:

$$K_{\text{BHATT}}(M_a, M_b) = \sum_i \exp\left[-\frac{(\mu_i^a - \mu_i^b)^T \Sigma_i^{-1} (\mu_i^a - \mu_i^b)}{8} \right] \tag{A-22}$$

Both the BHATT and WBHATT kernels are bounded, which is a major difference from the other kernels.

[A-1] C. H. You, K. A. Lee and H. Li, "GMM-SVM kernel with a Bhattacharyya-based distance for speaker recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 6, pp. 1310-1312, Aug 2010.

[A-2] W. Campbell, D. Sturim and D. Reynolds, "Support vector machines using GMM supervectors for speaker verification," *Signal Processing Letters*, 2006.

B. DERIVATION OF MINIMUM DIVERGENCE ESTIMATION

First we take a quick look at maximum likelihood estimation. The objective function is

$$L(V) = \sum_s \log P(X(s) \mid V) \tag{B-1}$$

By choosing $y(s)$ as the latent variable (which is a natural choice), we construct an auxiliary function:

$$\begin{aligned} Q(V,V') &= \text{constant} + \sum_s \int_{y(s)} P(y(s) \mid X(s),V) \log P(X(s), y(s) \mid V') \, \mathrm{d}y(s) \\ &= \text{constant} + \sum_s \int_{y(s)} P(y(s) \mid X(s),V) \log P(X(s) \mid y(s),V') \, \mathrm{d}y(s) \end{aligned} \tag{B-2}$$

Note the change in the term within the logarithm. This happened because (dropping the argument *s*)

$$\log P(X, y \mid V') = \log P(X \mid y,V') + \log P(y \mid V') \tag{B-3}$$

But the (prior) distribution of *y* doesn't depend on *V'*, so the second term gets absorbed by the "constant" term (remember that we're going to maximize *Q* in terms of *V'* only).

Now we make an "unnatural" choice about the latent variable – we choose the entire Vy, and denote it by *a*. By imitating Eq. (40), we can write out the auxiliary function:

$$Q'(V,V') = \text{constant} + \sum_s \int_a P(a \mid X,V) \log P(X, a \mid V') \, \mathrm{d}a \tag{B-4}$$

The meaning of this formula isn't obvious at first sight. First, the range of *a* doesn't cover the entire *CF*-dimensional space, so the probability density function $P(a \mid X,V)$ isn't well-defined. This doesn't matter, because we can treat the entire $\int_a P(a \mid X,V)[\cdot] \, \mathrm{d}a$ part as an expectation operator $E_{a|X,V}$ over the posterior distribution of *a*:

$$Q'(V,V') = \text{constant} + \sum_s E_{a|X,V}[\log P(X, a \mid V')] \tag{B-5}$$

Next let's study the term within the logarithm. We can decompose it in a similar way to Eq. (B-3):

$$\log P(X, a \mid V') = \log P(X \mid a,V') + \log P(a \mid V') \tag{B-6}$$

Now look at the first term: when *a* is given, the distribution of *X* is determined, and *V'* has no effect! Therefore this time it is the first term that gets absorbed by the "constant" term. The auxiliary function becomes

$$Q'(V,V') = \text{constant} + \sum_s E_{a|X,V}[\log P(a \mid V')] \tag{B-7}$$

If the range of *V'* were different from that of *V*, then $\log P(X, a \mid V')$ will be zero almost

everywhere over the posterior distribution of *a*, which is meaningless. It is required that the range of *V'* be identical to that of *V*, and this is why minimum divergence doesn't change the eigenvoice space.

Given that the columns of *V* and *V'* span the same subspace, we can let $V' = VJ$, where *J* is a small invertible matrix. Let's think about what $P(a \mid V')$ means: it means that a variable *y'*, whose prior distribution is the standard normal distribution, takes a value such that $V'y' = a = Vy$, which means $y = Jy'$. So, if we drop the notation of *a* and switch to *y*, the auxiliary function will become

$$Q'(V,V') = \text{constant} + \sum_s E_{y|X,V}[\log P(y = Jy')] \tag{B-8}$$

This is a cross-entropy from the posterior distribution of *y* to the distribution of Jy'. If we add the entropy of the posterior distribution of *y* itself (anyway it doesn't depend on *V'*), the auxiliary function becomes

$$Q'(V,V') = \text{constant} + \sum_s E_{y|X,V} \frac{\log P(y = Jy')}{\log P(y \mid X,V)} \tag{B-9}$$

which is then the negative of the KL-divergence between the two aforementioned distribution. Remember that the prior distribution of *y'* is the standard normal distribution. Maximizing the auxiliary function is equivalent to minimizing (the sum across all speakers of) the KL-divergence from the posterior distribution of *y* to the distribution of Jy' – this is why this updating procedure is called "minimum divergence estimation".

In Section 3.3.1.4 we have derived that the posterior distribution of *y* is (see Eq. (41)):

$$\mathsf{N}(E_1(s), G(s)) \tag{B-10}$$

And given that $y \sim \mathsf{N}(0, I)$, the distribution of Jy' is

$$\mathsf{N}(0, JJ^T) \tag{B-11}$$

The KL-divergence from a *k*-dimensional Gaussian distribution $\mathsf{N}_0(\mu_0, \Sigma_0)$ to another Gaussian distribution $\mathsf{N}_1(\mu_1, \Sigma_1)$, according to Wikipedia, is:

$$D_{\mathrm{KL}}(\mathsf{N}_0 \| \mathsf{N}_1) = \frac{1}{2}\left(\mathrm{tr}\left(\Sigma_1^{-1}\Sigma_0\right) + \left(\mu_1 - \mu_0\right)^T \Sigma_1^{-1}(\mu_1 - \mu_0) - \log\left(\frac{\det \Sigma_0}{\det \Sigma_1}\right) - k \right) \tag{B-12}$$

(The sum of) the KL-divergence to be minimized, dropping the terms that don't depend on *V'*, is

$$\Sigma D_{KL} = \frac{1}{2}\sum_s \left(\mathrm{tr}[(JJ^T)^{-1}G(s)] + E_1(s)^T (JJ^T)^{-1} E_1(s) + \log\det(JJ^T) \right) \tag{B-13}$$

Let $K = (JJ^T)^{-1}$, and we want to find the matrix K that minimizes this sum of KL-divergence. We will take its derivative to K. Here are some formulas of matrix calculus that come in handy:

$$\frac{\partial}{\partial A}\mathrm{tr}(AB^T) = B \qquad \frac{\partial}{\partial A}(x^T Ay) = xy^T \qquad \frac{\partial}{\partial A}\det A = (\mathrm{adj}\ A)^T = \det A \cdot (A^{-1})^T \quad \text{(B-14)}$$

where adj A is the adjugate of A. Also, since the matrices we're dealing with are all symmetric, we don't need to care about the transposes.

The derivative of ΣD_{KL} w.r.t K is

$$\begin{aligned}\frac{\partial}{\partial K}\Sigma D_{KL} &= \frac{1}{2}\frac{\partial}{\partial K}\sum_s \left(tr[KG(s)] + E_1(s)^T KE_1(s) - \log\det K\right) \\ &= \frac{1}{2}\sum_s \left(G(s) + E_1(s)E_1(s)^T - \frac{\det K \cdot K^{-1}}{\det K}\right) \\ &= \frac{1}{2}\sum_s \left(E_2(s) - K^{-1}\right)\end{aligned} \quad \text{(B-15)}$$

Set this derivative to zero, and we'll have

$$K^{-1} = JJ^T = \frac{1}{S}\sum_s E_2(s) \quad \text{(B-16)}$$

By Cholesky decomposition we can find J, and update V by $V \leftarrow VJ$. The update formulas for D and U are similar.

C. JFA SPEAKER ENROLMENT: JOINT ESTIMATION OF *x*, *y*, *z*

Analogous to Eq. (32), joint estimation of *y*, *x*, *z* entails calculating

$$[\hat{x};\hat{y};\hat{z}] = (I + [U\ V\ D]^T \Sigma^{-1} N [U\ V\ D])^{-1} [U\ V\ D]^T \Sigma^{-1} (F - NM) \tag{C-1}$$

The hard part of this is the matrix inversion $(I + [U\ V\ D]^T \Sigma^{-1} N [U\ V\ D])^{-1}$. To simplify the notation, let $W = [U\ V]$ and $S = \Sigma^{-1} N$, then the inversion becomes:

$$(I + [W\ D]^T S [W\ D])^{-1} = \begin{bmatrix} I + W^T SW & W^T SD \\ DSW & I + DSD \end{bmatrix}^{-1} \tag{C-2}$$

In this formula, *W* is a tall but think matrix, and *D* and *S* are large diagonal matrices. So the matrix to be inverted has a huge bottom-right corner that is diagonal.

We'll invert this matrix by row transformations. In the procedure, we'll introduce more letters to simplify the notation. We start with

$$\left[\begin{array}{cc|cc} I + W^T SW & W^T SD & I & 0 \\ DSW & I + DSD & 0 & I \end{array}\right] \tag{C-3}$$

Multiply the second row with $(I + DSD)^{-1}$, and subtract from the first row $W^T SD$ times the new second row:

$$\left[\begin{array}{cc|cc} I + W^T SW - W^T SD(I + DSD)^{-1} DSW & 0 & I & -W^T SD(I + DSD)^{-1} \\ (I + DSD)^{-1} DSW & I & 0 & (I + DSD)^{-1} \end{array}\right] \tag{C-4}$$

Let's look at the top-left block. Remember that $(I + A)^{-1}$ can be expanded as $I - A + A^2 - A^3 \cdots$, therefore

$$\begin{aligned} & I + W^T SW - W^T SD(I + DSD)^{-1} DSW \\ &= I + W^T SW - W^T SD(I - DSD + DSD^2 SD - DSD^2 SD^2 SD \cdots) DSW \\ &= I + W^T (S - SD^2 S + SD^2 SD^2 S - SD^2 SD^2 SD^2 S + SD^2 SD^2 SD^2 SD^2 S \cdots) W \\ &= I + W^T (I + SD^2)^{-1} SW \\ &= I + W^T (I + DSD)^{-1} SW \end{aligned} \tag{C-5}$$

Now let $P = (I + DSD)^{-1}$, then we have

$$\left[\begin{array}{cc|cc} I + W^T PSW & 0 & I & -W^T SDP \\ PDSW & I & 0 & P \end{array}\right] \tag{C-6}$$

Let $H = I + W^T PSW$, and multiply the first row with H^{-1}:

$$\left[\begin{array}{cc|cc} I & 0 & H^{-1} & -H^{-1}W^T SDP \\ PDSW & I & 0 & P \end{array}\right] \tag{C-7}$$

Subtract from the second row $PDSW$ times the first row:

$$\left[\begin{array}{cc|cc} I & 0 & H^{-1} & -H^{-1}W^T SDP \\ 0 & I & -PDSWH^{-1} & P + PDSWH^{-1}W^T SDP \end{array}\right] \tag{C-8}$$

Now the part on the right side of the dotted line is the inverse of we wanted to find. Its computation involves the inversion of two matrices ($I + DSD$ and H). The former is a diagonal matrix, and the second is small; the computation of both is tractable.

We can go on to calculate the estimates of *y*, *x* and *z*:

$$\begin{aligned} [\hat{x};\hat{y};\hat{z}] &= (I + [W\ D]^T S[W\ D])^{-1}[W\ D]^T \Sigma^{-1}(F - NM) \\ &= \begin{bmatrix} H^{-1} & -H^{-1}W^T SDP \\ -PDSWH^{-1} & P + PDSWH^{-1}W^T SDP \end{bmatrix}^{-1} \begin{bmatrix} W^T \\ D \end{bmatrix} \Sigma^{-1}(F - NM) \\ &= \begin{bmatrix} H^{-1}W^T(I - SDPD) \\ PD - PDSWH^{-1}W(I - SDPD) \end{bmatrix} \Sigma^{-1}(F - NM) \end{aligned} \tag{C-9}$$

To break this further down, we have:

$$[\hat{x};\hat{y}] = H^{-1}W^T(I - SDPD)\Sigma^{-1}(F - NM) \tag{C-10}$$

$$\begin{aligned} \hat{z} &= PD\Sigma^{-1}(F - NM) - PDSW[\hat{x};\hat{y}] \\ &= PD\Sigma^{-1}\{F - N(M + W[\hat{x};\hat{y}])\} \end{aligned} \tag{C-11}$$

We can see that Eq. (C-11) is exactly the same as Eq. (33). That means in both two-step estimation and joint estimation, the same formula is used to calculate *z* when *x* and *y* are known. However, in joint estimation, the matrix *D* also comes into play in the estimation of *x* and *y*.

LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS

BHATT	Bhattacharyya (a kernel function)
CA	correct accept
CDF	cumulative density function
CMN	cepstral mean normalization
CMS	cepstral mean subtraction
CR	correct reject
DSCC	delta-spectral cepstral coefficients
EER	equal error rate
EM	expectation-maximization
$F_{0.5}$	precision biased correct decision rate (an open-set evaluation criterion)
FA	false accept / false alarm
FFV	fundamental frequency variation
FR	false reject
GMM	Gaussian mixture model
GUMI	GMM-UBM mean interval (a kernel function)
HMM	hidden Markov model
HSCC	harmonic structure cepstral coefficients
IROSIS	integrated robust open-set speaker identification system
JFA	joint factor analysis
KL	Kullback-Leibler (divergence, also a kernel function)
L2	a kernel function
LLR	log-likelihood ratio
LMS	least mean squares
LP	linear prediction
MAP	maximum *a posteriori*
MD	minimum divergence
MFCC	Mel-frequency cepstral coefficients
MHEC	mean Hilbert envelope coefficients
ML	maximum likelihood
NIST	National Institute of Standards and Technology
PPMD	PRE_PEAK8_MFCC20_GAUSS300 + PRE_PEAK8_DSCC20(GAUSS300)_CMN (an acoustic feature)
RASTA	relative spectral (filtering)
RM40	an acoustic feature
ROSSI	robust open-set speaker identification
SCF	spectral centroid frequency

SCM	spectral centroid magnitude
SE	speaker error
SID	speaker identification
SNR	signal-to-noise ratio
SRE	speaker recognition evaluation
SVM	support vector machine
SWCE	sine-weighted cepstrum estimator (a type of multitapers)
UBM	universal background model
VAD	voice activity detection
WBHATT	weighted Bhattacharyya (a kernel function)
WMVDR	warped minimum variance distortionless response